SYSTÈME POUR L'ÉTIQUETAGE AUTOMATIQUE DE L'IMAGE

par

Gilles Daigle

mémoire présenté au Département de mathématiques et d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, mars 1999

Le 2 mars 1999 , le jury suivant a accepté ce mémoire dans sa version finale.

date

Président-rapporteur:	M. Béchir Ayeb Département de mathématiques et d'informatique
Membre:	M. Shengrui Wang Département de mathématiques et d'informatique
Membre:	M. Djemel Ziou Département de mathématiques et d'informatique
Membre:	M. Richard Lepage École de technologie supérieure

SOMMAIRE

La révolution informatique a mis à notre disposition un nombre phénoménal de bases de données de toutes sortes. Il est facile de rechercher un texte quelconque à l'aide de mots-clés, mais le domaine émergeant de l'audiovisuel pose un problème de taille. Une solution à ce problème est d'attacher une description textuelle à l'item problématique.

La présente recherche vise à la création d'un système pour l'étiquetage automatique des images. Ce domaine n'est pas entièrement nouveau, mais les systèmes proposés jusqu'ici n'en sont qu'à l'état expérimental. Nous nous sommes donc donnés comme critère de base que le système à développer devait paraître professionnel, être rapide et donner de bons résultats. Le but ultime est de créer un système de calibre commercial, mais il reste sans doute beaucoup de chemin à faire.

REMERCIEMENTS

Je voudrais premièrement remercier les professeurs Shengrui Wang et Béchir Ayeb pour la confiance qu'ils m'ont témoignée lors de ma demande d'inscription à la maîtrise et ce après une absence de 17 ans du milieu académique.

Je voudrais tout spécialement remercier mon directeur de recherche, Shengrui Wang, pour son aide, ses conseils, sa grande disponibilité et la confiance qu'il m'a accordée durant ces dernières années.

Je remercie Monique Daigle et Jean-Marc Groleau pour l'aide précieuse qu'ils m'ont donnée lors de la rédaction de ce mémoire. Merci aussi à tous mes amis qui m'ont soutenu et aidé durant ces dernières années.

Enfin, mes remerciements aux professeurs qui m'ont fait l'honneur de participer au jury de ce mémoire.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

La recherche d'images basée sur leur contenu émerge aujourd'hui comme un thème de recherche important ayant de nombreuses applications, notamment dans les bases de données multimédia et les bibliothèques numériques. En effet, avec la vaste quantité d'informations disponibles aujourd'hui, l'utilisation d'index et même d'outils de recherche devient cruciale. L'internet nous en offre un exemple typique. Nous n'avons jamais eu autant d'informations si facilement disponibles, mais trouver l'information *utile* devient un défi. De nouveaux outils de recherche et d'indexage apparaissent à tous les jours. Ces outils manipulent l'information textuelle pour faire leur travail. Cependant, l'internet devient un médium de plus en plus audiovisuel et indexer ce genre d'information est tout un problème.

Il y a beaucoup de recherche dans les systèmes d'extraction de données visuelles, mais beaucoup de ces systèmes exigent que la base de données visuelles et l'outil de recherche se retrouvent tout deux dans le même système informatique. Une meilleure approche serait de pouvoir utiliser les outils de recherches à base de texte que l'on retrouve partout sur l'internet dans le but d'intégrer la recherche d'images avec la recherche de textes. Pour cela, l'information visuelle doit être traduite en un type d'information textuelle

utilisable par les outils de recherche.

Présentement, cette traduction se fait par une personne qui soit écrit une description de l'image, soit en retire quelques mots-clés. Si l'on restreint le domaine aux images, une approche pour créer cette information automatiquement est d'étiqueter tous les éléments de l'image. Ceci nous donne une série de mots-clés se rapportant à l'image et même un pourcentage de l'image se rapportant aux mots-clés. Cette méthode a aussi l'avantage d'être moins subjective qu'une description de l'image par une personne.

Il faut souligner ici un point important: il y a un degré de subjectivité dans la création de mots-clés. Loin d'être un inconvénient, nous croyons que ceci est un avantage pour le genre de système que nous proposons. En effet, ceci veut dire que nous pouvons tolérer, sans trop de problème, un certain pourcentage d'erreur dans un système automatique d'étiquetage.

Nous proposons un système d'étiquetage à deux phases. Dans la première phase, l'usager instruit le système à l'aide d'une série d'images typiques. C'est la phase d'apprentissage. Dans la deuxième phase, le système peut être utilisé pour faire l'étiquetage d'un nombre illimité d'images. Le résultat est, pour chaque image, un fichier contenant les étiquettes que le système propose pour l'image. Ces fichiers d'étiquettes peuvent ensuite être utilisés par un autre système.

L'étiquetage automatique de l'image est un nouveau domaine de recherche et un article récent du MIT [27] a montré que des progrès pouvaient être accomplis avec un système qui sélectionne le meilleur modèle de représentation parmi plusieurs. Nous avons choisi, comme base, ce système qui semblait prometteur dans le but d'y apporter des améliorations et d'en faire un système à deux phases. La première partie de la présente recherche

fut donc de recréer le système décrit dans l'article. Dès le début de la programmation, il devint évident que la méthode de clustering[1] utilisée par le groupe du MIT demandait beaucoup de temps de calcul. Ceci peut limiter l'utilité du système d'étiquetage s'il est utilisé avec une base de données très grande.

Le premier défi fut donc d'améliorer la méthode de clustering. La méthode utilisée par le groupe du MIT est une modification de la méthode de Jarvis-Patrick [16]. Nous proposons ici une nouvelle méthode: l'arbre des distances. Cette méthode est beaucoup plus rapide que la méthode modifiée de Jarvis-Patrick et semble donner des résultats acceptables.

Le premier chapitre situe le projet dans le contexte de la recherche d'image. Le chapitre suivant présente le système d'une façon théorique et élabore sur les méthodes utilisées. Le troisième chapitre se concentre sur les algorithmes de clustering, une partie importante de tout système de classification. Le chapitre quatre traite des aspects pratiques du système que nous avons créé, ainsi que des résultats obtenus. Il se veut aussi un manuel de l'usager et donne les détails de l'implémentation qui seront utiles pour des recherches subséquentes.

[1]Nous utiliserons les mots **cluster** et **clustering** qui n'ont pas d'équivalent en français.

CHAPITRE 1

La recherche d'images

Bien que les ordinateurs existent depuis plus de 50 ans ce n'est que depuis les 10 dernières années qu'ils ont la puissance nécessaire pour permettre un accès direct à l'information audio-visuelle. En effet, le stockage et l'utilisation de ce type d'information dans un système informatique nécessitent beaucoup de mémoire, tant vive que secondaire, ainsi qu'une puissance de calcul non négligeable.

Nous assistons maintenant à une révolution numérique et tout ce qui est information deviendra numérique. Les disques de musique ne se vendent que de façon numérique maintenant. Plusieurs stations de radio ont une production entièrement numérique. Bientôt, la transmission sera aussi numérique. C'est la même chose pour la télévision. Des caméras numériques sont déjà en vente et remplaceront éventuellement les pellicules photographiques.

La technologie est en place pour créer de vastes quantités d'information audio-visuelle, mais il reste un gros problème à résoudre : comment avoir un accès intelligent à toute cette information. Dans la section qui suit nous abordons cette question pour en dégager les problèmes principaux et définir des principes de base. Nous appliquerons ensuite ces principes à la recherche de l'image qui pose un problème tout particulier.

1.1 Processus de classification

Le premier réflexe humain face à une grande quantité d'objets est de les classifier. L'exemple le plus immédiat d'un système de classification de grande quantité d'objets est sans doute la bibliothèque.

À la base de tout système de bibliothèque est le catalogage des objets. Pour permettre ce catalogage, il faut premièrement une représentation symbolique adéquate des objets en cause. Une façon de représenter un objet est d'en faire une description. Dans une bibliothèque numérique tout les objets ne sont eux-même qu'information et représentation symbolique. La description des objets d'une bibliothèque est donc de l'information sur l'information de la bibliothèque.

Dans le language de l'informatique, ce type d'information porte le nom de *méta-données*. Ainsi, le catalogue d'une bibliothèque ne contient que des méta-données. La création de bonnes méta-données est un problème complexe. Il n'y a pas de concensus sur la définition de bonnes méta-données. Les critères suivants se rapportant aux méta-données ont été identifiés [21] : accès, précision, disponibilité, contenu, consistance, coût, structure des données, facilité de création, facilité d'utilisation, économie, flexibilité, quantité, fiabilité,

standards, ... Il est clair que plusieurs de ces critères sont en conflit l'un avec l'autre et de toute façon ils ne sont pas absolus, mais dépendent de l'utilisation que l'on veut faire des méta-données.

Une question plus pertinente serait donc : qui va utiliser l'information et comment va-t-elle être utilisée? Le type d'utilisateur passe du novice à l'expert. Le novice a évidemment besoin d'un système facile à utiliser, mais ce qui le caractérise le plus c'est qu'il ne sait pas ce que contient la base de données. L'utilisation d'un outil de recherche sur le Web nous donne un exemple typique : le résultat de la recherche est *aucun item trouvé* ou bien *23,456 items trouvés*. Donc, la requête est souvent trop spécifique ou trop vague.

Une solution à ce problème est de permettre des requêtes de haut niveau afin de voir quel type d'information est disponible [11]. L'usager peut ainsi faire une recherche interactive, commencer par le vague et raffiner la demande dépendant de l'information disponible. Le type d'information nécessaire à ce genre d'accès doit être prévu lors de la cueillette des méta-données.

D'un point de vue pratique, il est important de considérer le temps d'accès d'une requête et la quantité de méta-données à transmettre devient un facteur important si la ligne de transmission est lente. Les données textuelles ont certainement un avantage dans ce cas.

En résumé, le type de classification utilisé n'est pas absolu, mais dépend de la façon dont on veut faire accès à l'information. Ceci va par la suite influencer le type de méta-données à cueillir.

1.2 Classification des images

Restreignons maintenant le sujet à la classification des images. L'image pose un problème tout particulier lors de la création des méta-données. Alors que pour les livres il existe maintenant des bases de méta-données (en plus de la page d'information pour le catalogage), ce type d'information n'existe pas pour les images. Présentement, le préposé au catalogage doit faire une classification et description (dans ses propres mots) de chaque image qui doit être cataloguée. En plus de la nature subjective de cette façon de procédé, ce travail est très lent et ainsi très peu d'images se retrouvent au catalogue.

Ouvrons une parenthèse ici pour parler du contexte de l'image. Si l'image provient d'une publication il est très possible que la description ou le sujet de l'image accompagne celle-ci. Il est aussi possible de faire une classification grossière si l'on connait l'auteur, la date, le lieu, l'évènement, etc. Dans une base de données d'images ce genre d'information doit accompagner l'image même si ce n'est que pour enregistrer les droits d'auteur. Dans ce qui suit, nous considérons donc les images en elle-même. L'information extraite provient directement de l'image.

Il existe une alternative à l'utilisation du texte dans la classification des images: c'est l'utilisation des propriétés de l'image elle-même. Quoi de plus naturel que de faire une demande par comparaison. L'usager peut ainsi demander une image qui ressemble à une autre, qui possède une couleur, des formes ou des textures spécifiques. Ce genre de demande pourrait être plus facile à formuler qu'une demande textuelle [9]. Comme l'emphase de ce genre de recherche se situe sur les propriétés de l'image, cette technique est appelé *recherche d'image basée sur le contenu.*

Plusieurs recherches sont en cours pour informatiser le processus de classification des images en se basant sur leur contenus. D'après Becker [3], le coeur de tout système de classification automatique des images est l'extraction des **caractéristiques**. Comme la plupart des systèmes actuels utilisent cette phase critique nous allons élaborer quelque peu sur ce sujet.

Il faut cependant souligner qu'il existe, dans le domaine des réseaux de neurones et même dans celui des algorithmes génétiques, des systèmes qui utilisent directement l'image pour accomplir leur classification et ne passent donc pas par la phase de l'extraction des caractéristiques. Par exemple, le système SHOSLIF [31] utilise des réseaux de neurones pour retrouver les images qui contiennent un objet quelconque.

1.2.1 Caractéristiques de l'image

Le dictionnaire [29] défini le mot **caractéristique** par *ce qui constitue un élément distinctif reconnaissable.* Certains mots qui si rattachent sont: attribut, indice, propriété, qualité, trait. Il faut donc en conclure que toute information qu'il est possible de tirer d'une image constitue une caractéristique de l'image.

Nous pouvons définir deux grandes classes de caractéristiques se rapportant à l'image : les caractéristiques de bas niveau et les caractéristiques de haut niveau. Les caractéristiques de **bas niveau** se rapportent à l'information qu'il est possible d'extraire directement de l'image: la couleur, la texture, la forme, la position, l'agencement des objets. Les caractéristiques de **haut niveau** se rapportent à l'interprétation des caractéristiques de bas niveau, c'est-à-dire, à une compréhension de l'image.

Parmi les caractéristiques de bas niveau, la couleur est l'une des premières qui nous vient à l'esprit. La couleur est, dans tout les cas, facile à extraire même dans les images où les objets sont difficiles à identifier. La couleur peut aussi servir de caractéristique globale de l'image. La distribution des couleurs dans une image est généralement représentée par un histogramme des valeurs d'intensités des couleurs primaires [9].

La texture est reconnue comme une propriété importante des images. Plusieurs systèmes de recherche d'images [34, 19, 10] utilisent la texture comme caractéristique principale pour l'extraction des images. Néanmoins, les opinions sont très variées sur la façon de représenter la texture. La section suivante va élaborer sur le sujet.

Les caractéristiques comme la forme, la position et l'agencement des objets sont plus difficiles à utiliser directement et se retrouvent généralement dans les systèmes d'indexation d'images qui utilisent les caractéristiques de haut niveau. Cependant, beaucoup de recherches se font sur l'extraction de ce genre de caractéristiques. La segmentation de l'image constitue la première étape importante (à part des préparatifs comme le lissage, etc.) à l'extraction des formes. Le but de la segmentation est de diviser une image en régions qui correspondent aux objets dans l'image ou aux parties des objets. Il est très difficile d'atteindre ce que Greenway [12] appelle la *segmentation idéale* et souvent l'image se trouve sous-segmentée ou sur-segmentée. Il existe maintenant un grand nombre de mécanismes de segmentation et faire un choix de méthode pour un problème particulier n'est pas évident. Un membre du groupe Neuro-Vision[1] en a fait une comparaison dans le but d'aider les chercheurs dans le choix d'une méthode [25].

Les systèmes de traitement d'images qui utilisent les caractéristiques de haut niveau

[1]Groupe Neuro-Vision : groupe de recherche en réseaux de neurones et traitement de l'image au sein du département d'informatique de l'Université de Sherbrooke.

sont plus complexes que les systèmes de bas niveau car il faut passer par le bas niveau pour se rendre au haut niveau (i.e. *il faut extraire les caractéristiques avant de les interpréter*). Il est cependant possible de créer des systèmes de recherche d'images très utiles en n'utilisant que des caractéristiques de bas niveau comme en fait foi la littérature sur le sujet.

Voici, selon notre avis, quelques-uns des avantages et désavantages des systèmes qui utilisent des caractéristiques de bas niveau à l'opposé des caractéristiques de haut niveau :

Système qui utilise des caractéristiques de bas niveau :

Avantages : Relativement facile à créer, peut être entièrement automatisé, usage intuitif (e.g. *image comme celle-ci*).

Désavantages : Pas de standard, ne permet pas de bases de données multiples, peut prendre beaucoup de place.

Système qui utilise des caractéristiques de haut niveau :

Avantages : Prend moins de place (*représentation plus compacte*), permet l'utilisation de systèmes à base de texte.

Désavantages : Plus complexe (*il n'existe pas encore de système entièrement automatisé*), demande intervention humaine pour classes, mot-clé, etc, souvent fait à la main.

1.2.2 Textures

Il est mentionné dans Haralick [13] qu'une approche formelle ou même une définition précise de la texture n'existe pas. Ceci est dû, je crois, au fait que le mot texture, qui nous vient du mot *textile* se rapporte à l'arrangement, à la disposition des éléments d'une matière. La texture n'est pas un concept visuel, mais plutôt un concept tactile. Plusieurs utilisent le concept d'*arrangement régulier* d'éléments visuels pour définir la texture [35].

Le mieux serait d'avoir un nouveau mot, mais si on applique le mot texture au domaine visuel il serait peut-être bon d'élargir sa définition. Dans le domaine de l'infographie et de la synthèse d'images on retrouve le concept de *peau* (skins) qui est la surface *texturée* qui recouvre le *squelette* de l'objet. Si on applique ce concept à la définition de la texture celle-ci devient l'aspect, la caractéristique visuelle de la surface d'un objet. Dans son livre sur la taxonomie des textures, Rao [28] définie la texture comme «l'apparence de la surface».

Ce qui complique le problème c'est que la texture visuelle d'un objet n'est pas invariante, mais dépend de la condition de prise de vue. Ainsi, la quantité et même la direction de l'illumination va influencer l'aspect d'un objet. Par sa *réflectance* l'objet lui-même va être plus ou moins influencé par la *qualité* de l'illumination (couleur de la lumière, homogénéité, etc.).

Il existe plusieurs facettes à l'analyse des textures [13], mais nous sommes tout spécialement intéressés ici à la classification. Le problème se réduit à ceci : étant donné une texture et un nombre fini de classes, déterminer dans quelle classe la texture appartient. Comme il est difficile de classifier directement les textures il faut extraire de celles-ci des

caractéristiques qui sont classifiables.

Parmi les grandes catégories de méthodes d'extraction des caractéristiques de textures, les méthodes statistiques sont certainement les plus utilisées et de plus se prêtent bien au processus de classification [35]. Les autres méthodes comme les méthodes géométriques, les méthodes basées sur un modèle et les méthodes de traitement de signal ne sont donc pas utilisées dans ce travail.

Les méthodes statistiques caractérisent l'image en fonction des variations d'intensités des points de l'image (voir 2.3.1). Ces méthodes statistiques peuvent elles-même être divisées en catégorie selon le nombre de points pris en considération. Avec les statistiques de premier ordre, chaque point de l'image est pris séparément. Les statistiques de second ordre s'intéressent aux points pris deux-à-deux.

Les statistiques de premier ordre se calculent généralement à partir d'histogrammes de couleurs ou de niveaux de gris et l'on y retrouve des mesures tel que la moyenne, la médiane, le mode, qui sont des mesures de l'illumination ou couleur typique de l'image, et la variance et l'écart-type qui donnent des mesures globale de dispersion des niveaux de gris ou de la couleur.

Pour mesurer les statistiques de second ordre, la matrice de co-occurence des niveaux de gris est utilisée (voir 2.3.2). L'information tirée de cette matrice donne une indication de l'homogénéité de l'image, de sa complexité, du contraste et de la direction des éléments à petite échelle [9].

1.3 Systèmes existants, travaux connexes

Il y a déjà quelques systèmes qui utilisent les caractéristiques de bas niveau sur le marché alors que les systèmes qui utilisent les caractéristiques de haut niveau n'en sont généralement qu'à l'état expérimental. Les systèmes qui suivent se retrouvent sur le commerce, ou du moins, sont en démonstration sur le Web. Ils utilisent un mélange de techniques qui sont en général de bas niveau :

Virage : Prétendent[2] être les premiers à présenter sur le Web un système de recherche d'images d'un vidéo. Le système utilise une série de technique pour faire un indexage automatique du film ce qui permet d'en visionner une section quelconque.

QBIC : [8] Produit commercial d'IBM, QBIC permet la recherche d'images qui ressemblent à une image clé. Dans le démo que l'on retrouve sur le Web, l'usager doit lui-même choisir les paramètres de recherche comme la couleur, la texture, l'agencement des couleurs, etc ...

Photobook : [26] Ce système, qui provient du MIT, utilise directement comme métadonnées une représentation compacte des caractéristiques de l'image. L'utilisateur peut ainsi interroger la base d'image en spécifiant l'allure de l'image, la forme générale d'un objet, la texture, etc.

Nous retrouvons dans la littérature plusieurs systèmes qui utilisent les caractéristiques de bas niveau pour faire l'indexation d'une base de données d'images. Chaque article attaque le problème sous un angle différent. En voici quelques exemples : Tamura

[2]Voir http://www.virage.com

[34] utilise les textures comme le contraste et la granularité et compare les résultats de l'indexation entre humain et machine. Manjunath [19] utilise des filtres de Gabor et des fonctions d'ondelettes pour classifier les textures. Swets [32] utilise l'image elle-même comme vecteur de caractéristiques de très haute dimension. Finalement, dans le système MARCO, Samet [30] utilise la forme de petits objets qui se retrouve dans l'image.

Le MIT Media Laboratory fait beaucoup de recherche sur les systèmes qui utilisent les caractéristiques de haut niveau et l'on retrouve plusieurs articles dans leur section de rapports techniques [10] [20] [27] [33].

Les buts d'un système de haut niveau peuvent être modestes. Le système décrit dans [10] fait un tri des images en deux catégories : scènes de ville ou de campagne selon que l'image contienne beaucoup ou peu de lignes droites. Un autre système [33] fait la même chose, mais avec des scènes intérieures ou extérieures selon les combinaisons de couleurs et de textures.

1.4 Conclusion

Pour résumer ce chapitre nous pouvons dire que la vague récente dans le problème de la recherche d'images est de faire premièrement l'extraction de caractéristiques de bas niveau pour ensuite soit utiliser directement l'information (méta-donné) dans un système de recherche d'images ou soit classifier cette information pour aboutir à un système de haut niveau où la reconnaissance de l'image (ou de ses parties) en devient le point saillant.

Nous avons choisi comme projet de maîtrise de créer un système pour l'étiquetage de

l'image pour les raisons suivantes :

- La création d'une liste de mots-clés pour chaque image permet l'utilisation des outils de recherche textuelle déjà existants. Donc, ce projet s'insère bien dans le cadre de la technologie actuelle.
- La précision absolue dans l'étiquetage n'est pas nécessaire. Les mots qui n'apparaissent qu'une fois peuvent être rejeté et le pourcentage des mots nous donne même une idée du contenu de l'image.
- Il est généralement facile de vérifier la performance d'un système d'étiquetage même si les choix de réponses sont parfois subjectifs.
- Ce genre de système utilise l'analyse des caractéristiques texturales qu'il est généralement facile d'extraire d'une image et de comparer. Comme il y a une infinité possible de caractéristiques et que certaines vont être plus utiles que d'autres dans le domaine de la reconnaissance, le problème n'est quand même pas trivial.
- Une équipe du MIT a démontré un système du même genre qui nous semblait prometteur. Nous ne sommes donc pas en terrain complètement inconnu.
- Ce projet, quelque peu démonstratif, posait un défi suffisant pour nous.

CHAPITRE 2

Principes de l'étiquetage

Dans ce chapitre, nous décrivons en détails les principes de base qui entrent dans la composition du système d'étiquetage de l'image que nous avons implémenté. Nous laissons pour le chapitre 4 les détails pratiques comme l'interface avec l'usager, la gestion de la banque d'images, la gestion des étiquettes, le stockage des résultats, etc.

Poser une étiquette sur une image (ou une imagette) est essentiellement un problème de classification. Généralement, un tel système fonctionne comme il est indiqué à la Figure 2.1. Une étude récente [24] de différents systèmes de classification qui fonctionnent selon ce principe donne un pourcentage de classification correcte qui se situe en général dans les 80 %. Cependant, il existe un problème important dans ce genre d'approche. Certaines caractéristiques peuvent être excellentes pour certains types d'images, mais donneront des résultats douteux pour un autre type d'images. Nous ne pouvons pas, tout simplement, additionner toutes les caractéristiques car ceci nous conduit à une dégrada-

tion des résultats, ce que l'on appelle communément la «malédiction de la dimension» (*curse of dimensionality*).

1. Extraction de certaines caractéristiques de l'objet.
2. Construction d'un vecteur de caractéristiques.
3. Apprentissage : création de **classes** de vecteur.
4. Recherche : comparaison d'un nouveau vecteur avec ceux des classes existantes.

Figure 2.1: Étapes usuelles pour la classification d'un objet.

Plusieurs chercheurs tentent de trouver **la** bonne combinaison de caractéristiques, mais ceci s'avère ne pas être la meilleure approche. Un principe de base qui motive les chercheurs en science pure veut qu'une bonne théorie n'ait peu, ou mieux encore pas du tout, de paramètres : c'est le principe du rasoir d'Occam[1]. Le même principe doit s'appliquer ici. L'intelligence (choix de caractéristiques, etc.) qui doit se retrouver dans le système de classification automatique ne doit pas être spécifiée entièrement par le concepteur, mais devrait se retrouver dans le système. Nous avons donc opté pour un système qui fait automatiquement un choix parmi plusieurs sous-ensembles de caractéristiques.

Pour faire un choix il faut évidemment avoir des critères de sélection. Ces critères correspondent ici au but du système d'étiquetage : avoir le meilleur étiquetage possible. Un bon étiquetage implique que la bonne étiquette se retrouve au bon endroit et que toutes les parties de l'image sont étiquetées. Nous avons donc deux critères : la quantité et la qualité. Comme la qualité est ici plus importante que la quantité, les buts de notre système sont, dans l'ordre : faire le moins d'erreur d'étiquetage possible, ajouter le plus

[1]William of Occam (1285-1349), philosophe anglais qui préconisait que le plus petit nombre possible d'hypothèses devrait être utilisé pour expliquer quoi que ce soit.

d'étiquettes possible (Figure 2.2). Ainsi, les expressions *maximiser* les résultats, faire le *meilleur* étiquetage ou trouver les *meilleurs* clusters se rapportent toujours à ces critères de sélection.

1. Faire le moins d'erreur d'étiquetage possible.
2. Ajouter le plus d'étiquettes possible.

Figure 2.2: Critère de sélection des clusters.

Avant d'entrer dans les détails, nous allons donner un aperçu rapide du système d'étiquetage de l'image tel que conçu. Cet aperçu va permettre de définir certains termes et concepts nécessaires à la compréhension des détails du système. Les sections qui suiveront vont reprendre chaque partie du système d'étiquetage et en expliquer le pourquoi et le comment.

2.1 Aperçu du fonctionnement

Pour les détails pratiques du fonctionnement du système le lecteur est prié de se reporter à la section 4.1. Comme il est mentionné dans l'introduction, le système est conçu pour fonctionner de deux façons: en mode d'apprentissage et en traitement par lots. Dans ce qui suit, nous ne parlerons pas du traitement par lots car toutes les opérations effectuées en traitement par lots sont aussi effectuées en mode d'apprentissage.

Au coeur du système il y a un mécanisme qui fait automatiquement un choix, pour chaque étiquette, parmi plusieurs sous-ensembles de caractéristiques de l'image. Ces caractéris-

tiques peuvent correspondre à la distribution des couleurs, au contraste, à la rugosité de l'image, à la directionnalité, à la périodicité, etc. Les caractéristiques extraites doivent être représentables dans un espace vectoriel car nous utilisons la distance euclidienne pour faire les comparaisons.

Définition : *Dans ce document, nous appelons* **modèle** *un sous-ensemble spécifique de caractéristiques.*

Généralement, le lecteur peut remplacer le mot **modèle** par l'expression *sous-ensemble de caractéristiques.* Cependant, le mot *modèle* peut être pris dans un sens plus large. Ainsi, *calculer une représentation du modèle* implique faire les calculs sur une imagette pour en extraire les caractéristiques se rapportant au modèle. Ceci nous donne un vecteur de caractéristiques se rapportant au modèle, ou plus simplement : vecteur du modèle. L'expression *exécuter un modèle* veut dire exécuter toutes les opérations du processus d'étiquetage pour un modèle spécifique.

Pour générer des étiquettes à partir d'une image (voir Figure 2.3), le système doit suivre les étapes suivantes :

Dans l'étape un, l'image doit être divisée en parties appelées imagettes. L'imagette est l'objet de base de notre système et les imagettes seront étiquetées manuellement par l'usager durant la période *d'apprentissage* ou automatiquement par le système durant la période *d'auto-étiquetage.*

Dans l'étape deux, l'usager ajoute les étiquettes sur les imagettes qui doivent servir d'étalonnage au processus d'étiquetage (étape 3c). L'algorithme d'étiquetage exige qu'il

1. Diviser une image en petites imagettes.
2. Ajouter aux imagettes les étiquettes spécifiées par l'usager.
3. Pour chaque modèle dans le système :
   (a) Calculer une représentation du modèle pour chaque imagette.
   (b) Créer un arbre hiérarchique de clusters avec les vecteurs du modèle.
   (c) Dégager individuellement les clusters de l'arbre, proposer des étiquettes où il en faut.
4. Comparer toutes les propositions des différents modèles (étape 3 a-c) pour trouver les *meilleures*.
5. Imprimer (ou afficher) les résultats.

Figure 2.3: Étapes pour le système d'étiquetage d'images.

y ait, dans un cluster, au moins deux imagettes qui ont reçus une étiquette identique de l'usager pour que la génération d'étiquette se fasse dans ce cluster. Cette règle existe dans le but de minimiser les erreurs car ce n'est pas bon de généraliser avec un seul exemple. L'usager doit donc s'assurer d'apposer au moins deux étiquettes sur chaque *objet* qu'il veut identifier.

L'étape trois se divise en trois sous-étapes qui se répètent pour chaque modèle (e.g. *chaque sous-ensemble de caractéristiques*), alors que l'étape quatre fera l'intégration de l'information ainsi obtenue. Si l'usager choisit de n'exécuter qu'un seul modèle, pour en vérifier l'efficacité par exemple, l'étape quatre sera sautée.

L'étape 3a consiste en la construction des vecteurs de caractéristiques tel que mentionné dans la Figure 2.1.

L'étape 3b consiste en la création d'un arbre hiérarchique de clusters. Les vecteurs de l'image en traitement sont combinés à tous les vecteurs enregistrés dans le système (vecteurs d'apprentissages).

L'étape 3c, consiste à extraire de l'arbre hiérarchique de clusters les plus gros clusters qui ne contiennent qu'un seul type d'étiquette spécifiée par l'usager. Toutes les imagettes non-étiquettées appartenant à ce cluster recevront ensuite cette étiquette.

La quatrième étape consiste en une sélection des clusters *gagnants* (selon les critères de la Figure 2.2) parmi tout ceux qui ont été générés dans l'étape trois. Cette sélection n'est pas globale, mais se fait pour chaque étiquette. L'étape consiste en la sélection d'un minimum de clusters qui regroupent toutes les imagettes qui possèdent l'étiquette spécifique.

Enfin, dans la dernière étape, les résultats pour chaque image sont sauvegardés dans des fichiers se rapportant à l'image et selon le cas, les étiquettes affichées à l'écran seront mises à jour.

Dans les sections qui suivent, certaines de ces étapes seront reprises avec plus de détails.

2.2 Création des imagettes

Il y a deux façons de diviser une image en imagettes. L'application d'une grille sur l'image nous permet de la découper en surfaces régulières. Les avantages principaux sont la facilité d'accomplir ce découpage, le fait que chaque imagette est rectangulaire et de grandeur

uniforme, ce qui facilite l'opération d'extraction des caractéristiques et l'utilisation d'un système de coordonnées qui permet de repérer chaque imagette et d'appliquer les étiquettes au bon endroit. La deuxième méthode consiste à utiliser l'information dans l'image elle-même pour effectuer le découpage (*segmentation* de l'image). Par exemple, l'utilisation d'un détecteur de contour nous permet de diviser l'image de façon à ce que chaque imagette représente un objet dans l'image. Il est aussi possible d'utiliser les textures elles-même, comme il est démontré dans [20], pour faire la segmentation de l'image.

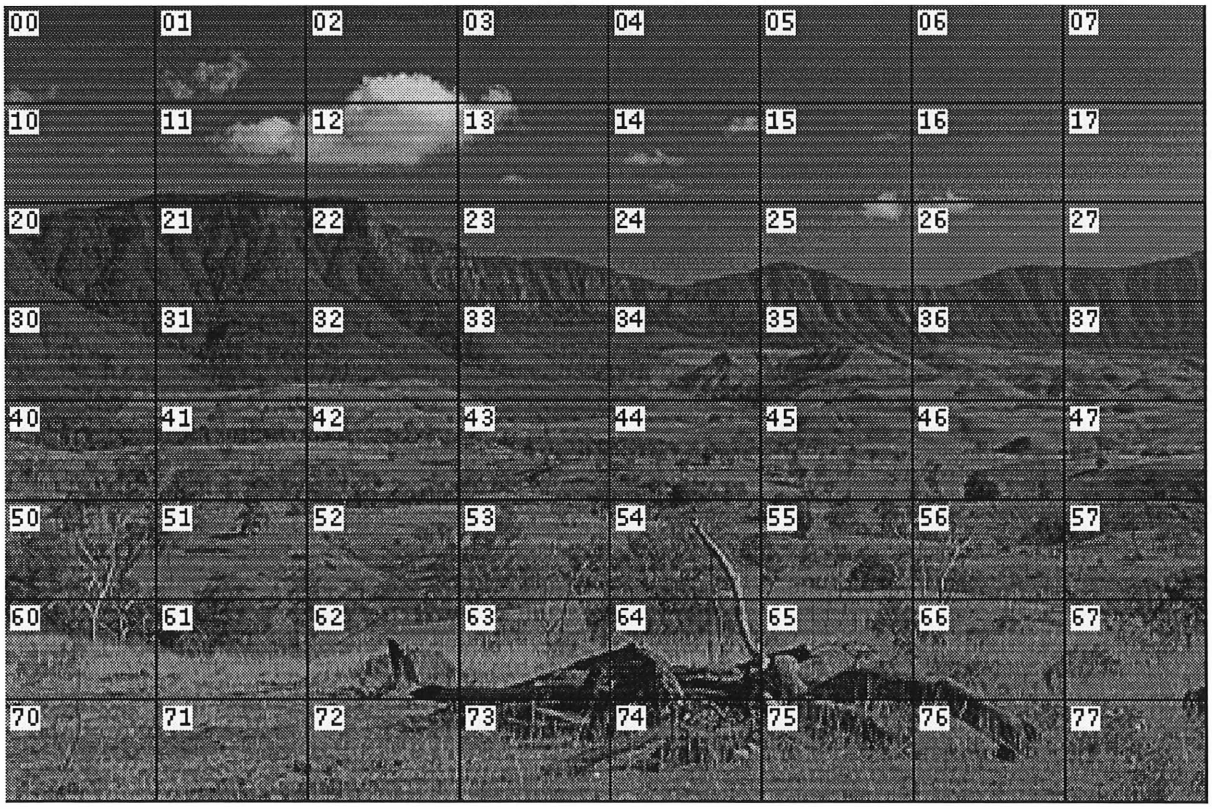


Figure 2.4: Grille d'imagettes avec coordonnées.

Pour la première phase de ce projet nous utilisons la première méthode à l'aide d'une grille rectangulaire (voir Figure 2.4). La sélection du nombre de rectangles dans la grille, et ainsi la grandeur de l'imagette, est un problème en soi. Les imagettes doivent être assez grande pour permettre l'application de modèles de textures. Si l'imagette est trop

grande elle va contenir plusieurs objets et sera difficile à étiqueter. D'un autre côté, s'il y a trop d'imagettes cela va nuire à l'efficacité du mécanisme de classification. Nous utilisons, dans ce travail, une grille de 8 par 8.

2.3 Extraction des caractéristiques

Comme il a été mentionné, un modèle est une collection de caractéristiques. De nouveaux modèles peuvent donc facilement être créés et incorporés au système. Le système fait automatiquement le choix des meilleurs modèles pour chaque situation ce qui fait que les modèles inefficaces n'auront pour effet que d'utiliser les ressources informatiques. Les ressources, en mémoire et temps de l'UCT, qu'un modèle utilise ne doivent pas être ignorées. Un modèle sera meilleur qu'un autre si, à part égale, il utilise moins de ressources. La qualité principale d'un modèle est de permettre un bon clustering des imagettes (selon les critères de la Figure 2.2). Cependant, nous favorisons les modèles *économiques*, c'est-à-dire, les modèles qui prennent peu de ressources informatiques.

2.3.1 Encodage de l'image

Pour bien comprendre l'extraction des caractéristiques il est bon de faire un rappel sur les différents formats d'encodage de l'image.

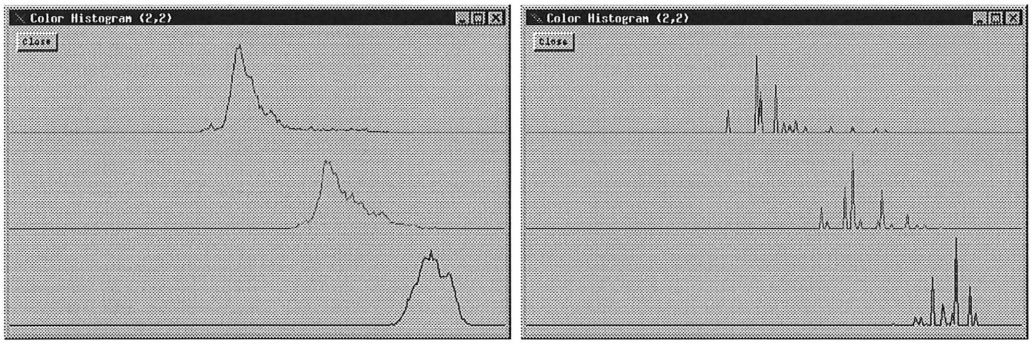
L'oeil humain possède des détecteurs de couleurs (les cones) qui sont sensible à trois couleurs bien précises: le rouge, le vert et le bleu. Toutes les couleurs que nous percevons

sont formées en additionnant ces trois couleurs *primaires* de façon différentes. Un écran couleur nous donne les images couleurs en utilisant les mêmes trois couleurs primaires. En anglais, ces couleurs sont *red, green, blue* ce qui nous donne l'expression: format RGB. Dans un ordinateur, ces trois couleurs primaires son représentées par trois nombres de 8 bits ce qui donne des valeurs de 0 à 255 pour chaque couleur. Un écran couleur peut donc représenter un maximum de $256^3 = 16,777,216$ couleurs différentes.

Quand il y a peu de lumière, les couleurs disparaissent et nous voyons en *noir et blanc*. L'oeil humain contient des *bâtonnets* qui sont sensibles aux faibles intensités de lumière mais qui ne distinguent pas les couleurs. Dans un ordinateur, les éléments d'une image en *noir et blanc* dites à niveaux de gris sont représentés par un seul nombre de 8 bits. Pour passer de la couleur au niveau de gris il suffit d'appliquer la formule suivante : $I = .33R + .5G + .17B$ [14]. Cette formule s'explique par le fait que l'oeil n'a pas la même sensibilité aux trois couleurs de base.

L'image elle même est généralement représentée dans un ordinateur par une matrice ou chaque position de la matrice représente un point de l'image. Ces points sont appelés **pixel**, mot qui vient de l'expression *picture element*. Une image moyenne nécessite environ un million d'octets alors qu'une image à haute définition va en nécessité plusieurs millions. Il existe plusieurs formats pour enregister les images en mémoire secondaire. Le format **PPM**, qui est sans doute le plus simple, ne fait que stocker la matrice de l'image et les informations sur ses dimensions. Le format **GIF** ne garde que les 256 couleurs les plus représentatives de l'image et fait ensuite une compression des pixels identiques. Le format **JPG** fait une compression de l'image entière mais l'usager peut choisir la précision de la compression. Il faut noter qu'il y a perte d'information avec l'utilisation des formats GIF et JPG.

Mentionnons ici l'un des problèmes rencontrés avec ces deux différents formats d'encodages d'images que l'on retrouve sur le Web. L'histogramme des couleurs de la Figure 2.5 démontre ce type de problème. Une image GIF est composée de 256 couleurs et l'histogramme correspondant va avoir plusieurs petits pics. Une image couleur JPG va avoir un histogramme plus lisse. Pour comparer des images provenant de différentes sources un pré-traitement des images devrait donc avoir lieu.



a. JPG (16M couleurs) b. GIF (256 couleurs)

Figure 2.5: Deux histogrammes RGB de la même imagette qui montre la différence entre les formats JPG et GIF.

Notre système devrait aussi accepter des images de différentes grandeurs. Il en résulte que les imagettes peuvent avoir différentes grandeurs et il est important de normaliser les résultats lors du calcul des caractéristiques. Par exemple, lors du calcul d'un histogramme, le total de toutes les valeurs devrait être égal à un.

2.3.2 Modèles utilisés

Nous allons maintenant décrire les modèles utilisés jusqu'ici avec leurs avantages et désavantages. Jusqu'ici, nous n'avons utilisé que des statistiques de premier ordre et de deuxième ordre. Les résultats obtenus avec ces modèles se retrouvent à la fin du chapitre 4.

Pour les statistiques du premier ordre, nous avons choisi de travailler avec la couleur car celle-ci joue un rôle important dans la reconnaissance et l'utilisation de la couleur ne pose pas trop de problèmes. Pour les statistiques du deuxième ordre nous utilisons des images à niveaux de gris, ce qui réduit de beaucoup le nombre de calcul à faire.

- **Histogramme des couleurs RGB**

Ce type de modèle fut utilisé originellement pour plusieurs raisons : il est facile à calculer et à visualiser et il génère beaucoup de données pour tester les limites des programmes. Comme chaque pixel d'une image couleur est un vecteur de trois composantes, un histogramme complet d'une telle image comporte 768 composantes. Pour une petite image, l'histogramme peut prendre la même quantité de mémoire que l'image.

- **Histogramme de teinte**

L'histogramme de teinte utilise la première composante du vecteur HSV[2]. Deux raisons ont incité l'utilisation de ce modèle : il utilise moins de données que l'histogramme RGB et il est possible que la teinte, utilisée seule, puisse donner de bons résultats.

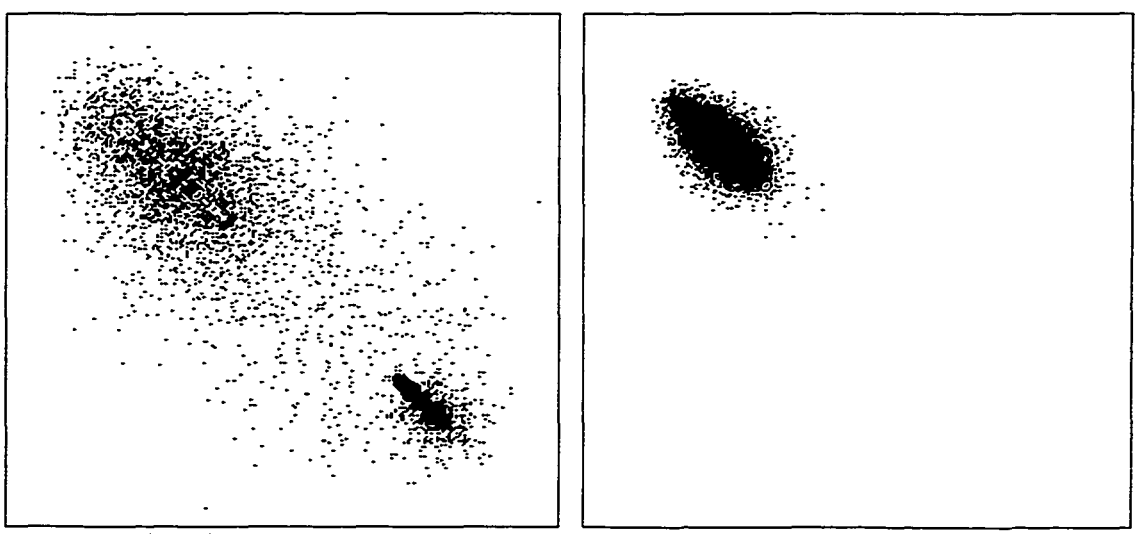
- **Moyenne des couleurs**

Pour chaque couleur, la moyenne et l'écart type sont calculées. Ceci nous donne un vecteur de 6 composantes, ce qui est très économique en mémoire et temps de calcul. Nous voulions voir si un modèle aussi sommaire pouvait donner des résultats acceptables.

- **Matrice de co-occurence**

[2]Pour «Hue, Saturation, Value». Autre façon de représenter les pixels couleurs dans un ordinateur.

Les statistiques du deuxième ordre sont calculées à partir de la matrice de co-occurrence des niveaux de gris. Une matrice de co-occurrence spécifie la fréquence P_{ij} avec laquelle deux pixels séparés d'une distance d et un angle θ apparaissent dans l'image, l'une avec un niveau de gris i et l'autre j. Deux pixels voisins ont une distance de un, s'ils sont séparés par un pixel la distance est deux, etc. L'angle peut varier de 0° à 180° mais les valeurs communes sont de 0° pour l'horizontale et 90° pour la verticale. Il faut noter que de telles matrices sont symétriques car on utilise l'angle entre deux points et non la direction. Il y a deux exemples dans la Figure 2.6, les deux ont un angle de 0° et une distance de 1. Les imagettes se retrouvent à la Figure 2.4; l'imagette (0, 0) représente du ciel et l'imagette (7,6) du gazon. La Figure 2.6 devrait être prise comme une matrice de 256x256 éléments où un point noir représente une valeur non-nulle.



a. Ciel (0, 0) b. Gazon (7, 6)

Figure 2.6: Matrice de Co-occurrence des niveaux de gris. Les points noirs représentent des valeurs non-nulles. Les axes représentent les niveaux de gris de deux pixels comparés.

Certains auteurs préconisent l'utilisation directe de la matrice de co-occurence (voir Haralick [13] page 460). Comme il y a 256 niveaux de gris dans une image standard ceci donne une matrice de co-occurrence assez large. Il est possible de réduire le nombre de

niveaux de gris pour avoir une matrice plus petite.

Dans ce modèle nous utilisons directement la matrice, mais quantifiée à 16 niveaux de gris, ce qui nous donne un vecteur de caractéristiques de 256 composantes. Nous pouvons aussi créer plusieurs modèles en changeant les paramètres de distance d et angle θ de la matrice.

- **Caractéristiques de la matrice**

Très souvent, la matrice n'est pas utilisée telle quelle. Voici quelques exemples de caractéristiques que nous pouvons extraire de la matrice:

Uniformité de l'énergie	$\sum_{ij} P_{ij}^2$
Entropie	$-\sum_{ij} P_{ij} \log P_{ij}$
Fréquence maximum	$\max_{ij} P_{ij}$
Contraste	$\sum_{ij} (i-j)^2 P_{ij}$
Tendance au clustering	$\sum_{ij} (i+j-2\mu)^2 P_{ij} \log P_{ij}$ où $\mu = \sum_{ij} i P_{ij}$

L'uniformité de l'énergie nous donne le même genre de renseignement que l'entropie, c'est à dire une mesure de la complexité de l'image. Les autres caractéristiques nous donne une idée du contraste de l'image, et de la tendance au regroupement. Nous avons créé un modèle unique en regroupant toutes ces caractéristiques, mais une étude plus poussée pourrait être faite en utilisant chaque combinaison de ces caractéristiques.

2.4 Regroupement d'imagettes (clustering)

L'application d'un modèle aux imagettes nous donne une série de vecteurs. Nous en sommes maintenant au stade du clustering, mais plusieurs problèmes se posent lors de la sélection du type d'algorithme de clustering que nous devons utiliser. Avant d'entrer dans les détails des algorithmes, nous allons donc expliquer pourquoi nous utilisons un algorithme de clustering qui nous donne un arbre hiérarchique de clusters (étape 3b), pour ensuite devoir dégager individuellement les clusters de l'arbre (étape 3c), plutôt que d'utiliser un algorithme de clustering qui nous donnerait directement une série de clusters.

Avant de continuer, il faudrait s'assurer de la définition de certain mots. L'opération de **clustering** est équivalente à séparer un ensemble d'éléments en sous-ensembles disjoints de telle sorte que tous les éléments d'un sous-ensemble jouissent d'une propriété commune [15]. Donc, un **cluster** est un sous-ensemble d'éléments. Il faut noter que l'**arbre hiérarchique** ne contient pas de clusters. Il est cependant possible de couper l'arbre en morceaux (en branches) et chaque morceau serait ainsi un cluster.

Pour illustrer les idées qui vont suivre, nous utiliserons le petit arbre hiérarchique représenté à la Figure 2.7. Les feuilles avec un «?» sont sans étiquettes alors que celles avec des lettres indiquent un étiquetage par l'usager. Dans cet exemple, deux clusters seront dégagés de l'arbre durant la deuxième phase du clustering: le noeud 2 et la feuille 12. La feuille 12 est inutile car elle est seule et déjà étiquetée. Le noeud 2 contient deux imagettes non étiquetées qui pourront donc recevoir l'étiquette «A».

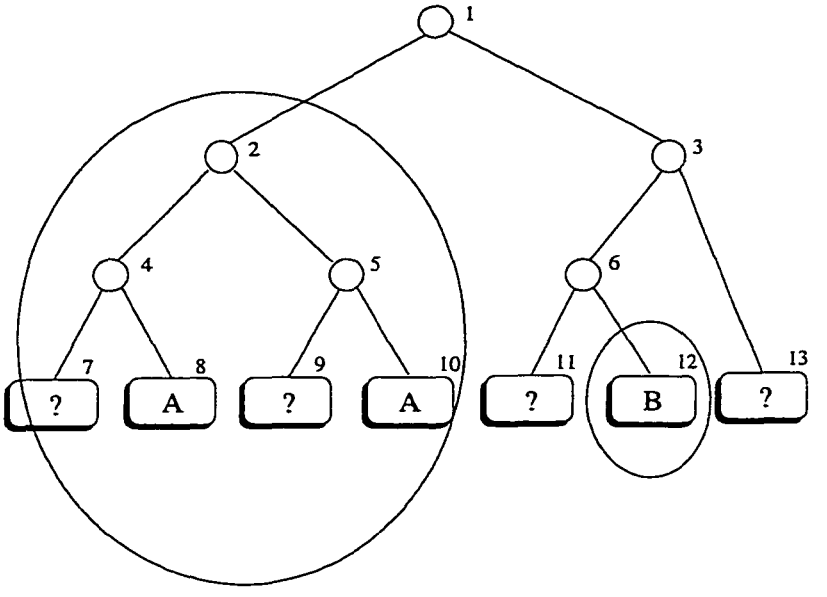


Figure 2.7: Petit arbre hiérarchique étiqueté. Les noeuds et feuilles sont numérotés pour les besoins du texte. Les clusters à dégager sont encerclés.

2.4.1 Choix d'algorithme de clustering

Les algorithmes de clustering non-hiérarchiques peuvent être divisés en deux classes selon que l'on doive spécifier, comme argument, le nombre de clusters à générer ou pas. Dans le problème qui nous occupe nous connaissons le nombre d'étiquettes dans le système, mais celui-ci n'est pas nécessairement égal au nombre de clusters à créer. En effet, l'étiquetage est une opération subjective qui ne correspond pas nécessairement à un simple clustering.

Voici un exemple pour illustrer cette idée : Dans une image, le ciel peut être bleu ou même orange, le lac peut être bleu ou blanc s'il y a de l'écume, la texture des arbres varie d'un endroit à l'autre, etc. Une fois l'opération de clustering terminé, le système peut se retrouver avec deux clusters ciel: un bleu et un orange. L'usager, lui, peut décider qu'il n'y a qu'un ciel et n'apposer qu'une étiquette de type ciel. Nous pouvons voir ici

qu'une étiquette spécifique se retrouvera dans plusieurs clusters séparés dépendant des variations dans le type d'objets en question. En d'autres mots, il n'y a pas de concordance une-à-une entre les étiquettes et les clusters.

Examinons un autre sénario: D'un autre coté il est possible, car aucun algorithme de clustering n'est parfait, qu'un beau gros cluster *ciel* se retrouve avec un étiquette *lac* à l'intérieur. Il serait dommage de rejeter le cluster au complet. Une meilleure solution serait de séparer le cluster en trois parties : deux clusters ciel et un petit cluster lac.

L'utilisation d'un arbre hiérarchique de clusters résout ces problèmes. Premièrement, pour créer l'arbre il n'est pas nécessaire de spécifier le nombre de clusters voulus, l'arbre contient un continuum de clusters. Deuxièmement, une fois l'arbre complété, nous pouvons utiliser une technique (voir 2.4.3) qui fait appel à toutes les informations que nous avons sur les imagettes pour sélectionner les clusters qui ont les caractéristiques et la grosseur qui conviennent.

2.4.2 Arbre hiérarchique de clusters

La création d'un arbre hiérarchique de clusters demande beaucoup de calcul et c'est l'un des domaines où notre approche diffère substantiellement de la méthode de clustering préconisée par l'équipe du MIT [27]. Ce problème de temps de calcul se fait surtout sentir lors de la phase d'apprentissage qui nécessite une interaction directe entre l'utilisateur et le système.

L'équipe du MIT utilise une adaptation de la méthode des voisins-communs telle que

décrite dans [15]. Cette méthode, créée par Jarvis et Patrick, fut l'une des premières à être publiée (en 1973) et est donc utilisée couramment. Avec cette méthode, deux points (e.g. imagettes) sont regroupés ensemble s'ils ont un certain nombre de voisins en commun. Il est à noter que c'est une méthode **non** hiérarchique. Il est possible de la transformer en méthode hiérarchique en modifiant le nombre de voisins et en regroupant les clusters ainsi formés, mais ceci multiplie le nombre de calculs à faire.

L'application de cette méthode de clustering au système d'étiquetage d'images soulève deux problèmes principaux :

- La distance entre toutes les imagettes doit être calculée (temps de calcul $O(N^2)$, où N est le nombre d'imagettes).
- De nouvelles imagettes ne peuvent être ajoutées sans refaire tous les calculs.

L'équipe du MIT a ignoré ces problèmes en spécifiant que la création de l'arbre doit être faite séparément dans une phase de pré-traitement.

Pour résoudre ces problèmes, nous avons créé une nouvelle méthode de clustering hiérarchique que nous appelons **l'arbre des distances**. L'arbre des distances est un arbre binaire où toutes les imagettes similaires se retrouvent ensemble, en tant que feuilles, sur la même branche. La Figure 2.8 nous montre une partie d'un arbre des distances. Les grappes de feuilles peuvent clairement être identifiées. L'algorithme utilisé pour créer l'arbre est récursif.

L'arbre des distances résout les deux problèmes principaux de la méthode de clustering des voisins-communs en permettant d'ajouter chaque imagette une à une et en faisant les

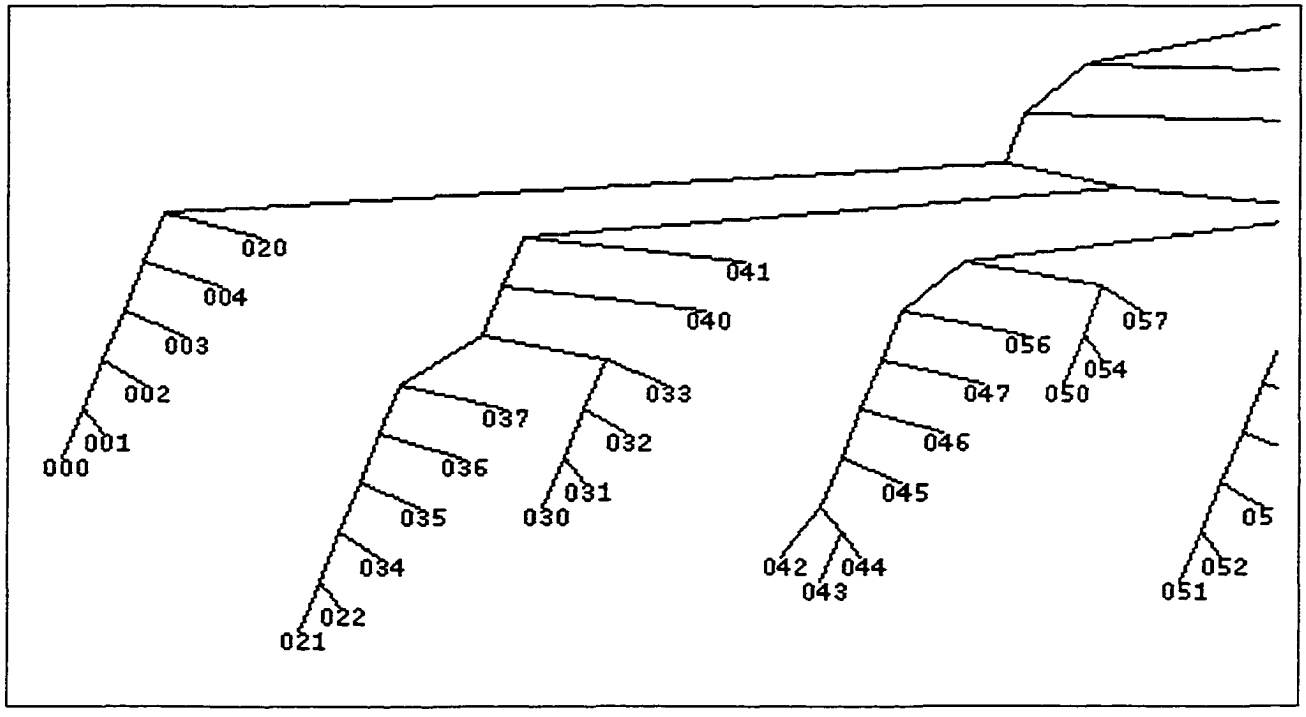


Figure 2.8: Section d'un arbre des distances. Chaque nombre correspond à une imagette de la fig. 2.4.

comparaisons des nouvelles imagettes sur les imagettes d'un arbre binaire et non entre toutes les imagettes dans la base de données. Une nouvelle imagette peut être ajoutée avec $\log_2(N)$ comparaisons, où N est le nombre d'imagettes dans l'arbre, car avec chaque comparaison la moitié de l'arbre est éliminé des comparaisons. D'un autre coté, l'ordre de présentation des imagettes semble important et le clustering avec l'arbre des distances ne semble pas aussi bon qu'avec la méthode de Jarvis-Patrick. Nous avons donc sacrifié quelque peu la précision du clustering pour un gain appréciable de vitesse d'exécution.

Vu l'importance de ce sujet dans notre recherche, le chapitre 3 se consacre entièrement à l'étude de ces algorithmes de clustering.

2.4.3 Extraction des clusters et étiquetage

Une fois en possession de l'arbre hiérarchique de clusters nous pouvons utiliser l'information que nous donne les étiquettes connues, qui sont attachées aux imagettes, pour extraire de l'arbre les clusters désirés. Les étiquettes connues sont évidemment les étiquettes spécifiées par l'usager. Dans ce qui suit, nous parlerons de deux types d'étiquettes: les étiquettes spécifiées par l'usager, et les étiquettes ajoutées par le système. Pour ne pas confondre ces deux types d'étiquettes nous utiliserons la notation suivante:

υ-étiquette: étiquette spécifiée par l'usager.

σ-étiquette: étiquette ajoutée par le système.

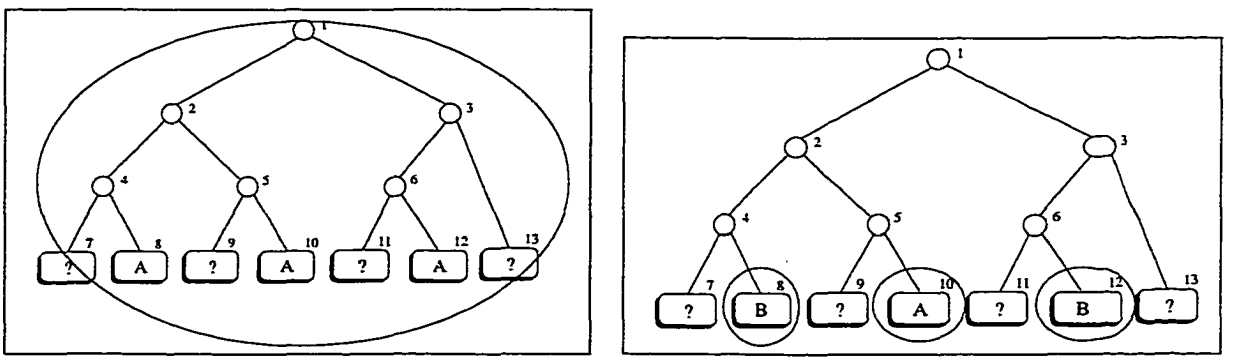
La Figure 2.9 donne une liste de critères à utiliser lors de l'extraction des clusters. Ces critères sont commandés par le but à obtenir et par le bon sens.

- Les imagettes des clusters ne doivent contenir qu'un seul type d'υ-étiquette.
- Les clusters doivent être aussi gros que possible.
- Les clusters qui ne contiennent que des imagettes non étiquetées sont inutiles car il est impossible de leur ajouter des σ-étiquettes.
- Dans le but de ne pas trop généraliser l'σ-étiquetage, un cluster doit contenir au moins deux imagettes υ-étiquetées pour que les imagettes non étiquetées reçoivent une σ-étiquette.

Figure 2.9: Critères dans la sélection des clusters.

Pour illustrer le problème, revoyons la section d'arbre reproduite à la Figure 2.7. Nous pouvons voir, en utilisant cet exemple et les critères de sélection, que l'arbre devrait être

divisé en deux clusters : la branche 2 et la feuille 12. Si la feuille B était un A, la section d'arbre au complet ne formerait qu'un seul cluster (Figure 2.10a). Par ailleurs, si un A devenait un B, nous aurions 3 clusters composés des feuilles 8, 10, 12 (Figure 2.10b).



a. Un seul cluster. b. Clusters feuilles.

Figure 2.10: Deux exemples de détermination des clusters.

Il s'avère qu'accomplir l'extraction des clusters en utilisant les critères de la Figure 2.9 est une tâche relativement simple, qui consiste à visiter systématiquement toutes les parties de l'arbre et à couper les branches les plus larges qui contiennent des étiquettes identiques. Chaque branche coupée sera ainsi un cluster. C'est ce que nous expliquons dans la prochaine section.

Algorithme d'extraction

L'algorithme d'extraction des clusters (Figure 2.11) est récursif et fait un parcours linéaire de l'arbre de gauche à droite. C'est donc un algorithme rapide de complexité $O(N)$, où N est le nombre de noeuds et feuilles dans l'arbre. L'algorithme reçoit comme argument la racine de l'arbre et le nom de l'étiquette à extraire. Le résultat final est une liste de clusters.

```
Variables Globales: S, bonne_étiquette
Extrait_racine( racine, étiquette )
        bonne_étiquette = étiquette
        si Extrait_branche( racine ) == bon alors imprime( S )
fin
Extrait_branche( branche )
        Variables: gauche, droite
        si branche == feuille alors
                si feuille->étiquette == bonne_étiquette retourne bon
                si feuille->étiquette != bonne_étiquette retourne mauvais
                si feuille->étiquette == pas_étiquette retourne neutre
        gauche = Extrait_branche( branche->gauche )
        droite = Extrait_branche( branche->droite )
        si gauche == bon alors
                si droite == bon { S = branche; retourne bon }
                si droite == mauvais { imprime( S ) ; retourne mauvais }
                si droite == neutre { S = gauche; retourne bon }
        si gauche == neutre retourne droite
        si gauche == mauvais alors
                si droite == bon alors S = droite
                retourne mauvais
fin
```

Figure 2.11: Algorithme récursif d'extraction des clusters.

Quelques explications sont de mise. La variable globale S contient la solution trouvée à date. Une solution peut être imprimée (avec imprime(S)) ou être remplacée par une

solution plus générale. Les valeurs de retour de l'algorithme sont : bon, mauvais, neutre selon que les branches contiennent de bonne, mauvaise ou pas d'étiquette. Une bonne étiquette est évidemment une étiquette qui est conforme à l'étiquette à extraire.

Le tableau 2.1 résume les décisions de l'algorithme et il est peut-être plus facile à comprendre. Chaque carré donne l'action à accomplir et le code retour pour chaque possibilité. Les points suivants indiquent comment utiliser le tableau :

- Si c'est une feuille, utiliser la ligne feuille du tableau.

- Si c'est la racine, traiter comme une branche ensuite utiliser la ligne racine du tableau.

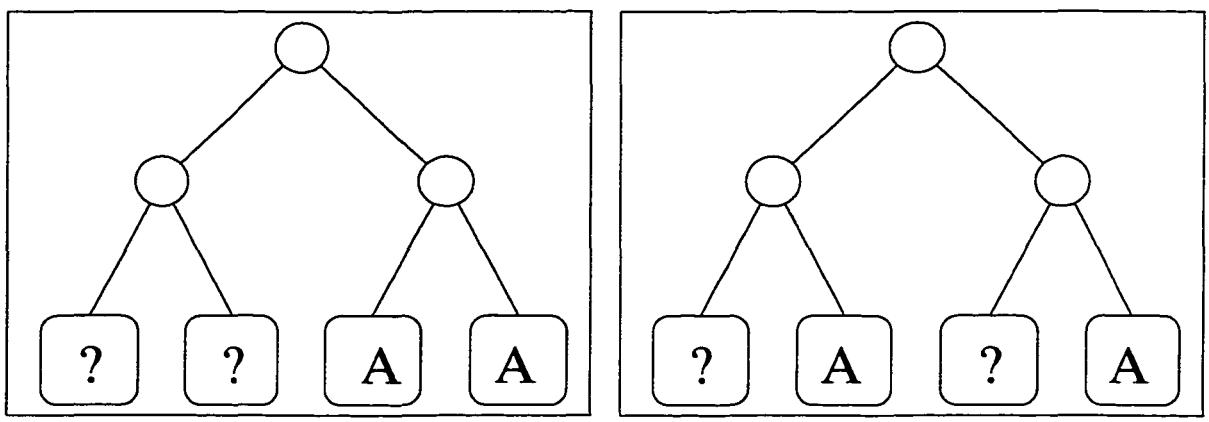
- Si c'est une branche, faire un appel récursif pour chacun des deux cotés de la branche. Comme chaque coté peut retourner trois valeurs nous avons 9 possibilités. Utiliser les lignes bon, neutre, mauvais du tableau pour le coté gauche de la branche et les colonnes pour le coté droit.

	bon	neutre	mauvais
feuille	S=feuille; bon	rien; neutre	rien; mauvais
bon	S=branche; bon	S=gauche; bon	imprime(S); mauvais
neutre	S=droite; bon	rien; neutre	rien; mauvais
mauvais	imprime(S); mauvais	rien; mauvais	rien; mauvais
racine	imprime(S); fin	rien; fin	rien; fin

Tableau 2.1: Résumé des décisions de l'algorithme d'extraction.

Utilisation de l'arbre binaire

La décision d'utiliser des arbres binaires a beaucoup simplifié la programmation du système tout en augmentant la vitesse d'exécution, mais certaines anomalies d'étiquetage nous laissent penser que ce n'était peut-être pas le meilleur choix.



a. Pas de propagation. b. Propagation des étiquettes.

Figure 2.12: Problèmes avec l'utilisation d'un arbre binaire.

La Figure 2.12 nous montre deux arbres qui, bien que presque identiques, ne seront pas étiquetés de la même façon par l'algorithme d'ajout d'étiquettes. Dans l'arbre a) les étiquettes ne seront pas propagées alors que dans l'arbre b) elles le seront. Ce genre de problème ne se poserait pas dans un arbre non-binaire. Nous pouvons en conclure qu'en utilisant un arbre binaire nous avons diminué le pourcentage des imagettes qui seront étiquetées.

2.5 Sélection globale des clusters

Chaque série de clusters (une pour chaque modèle) créée à la section précédente est en soi une solution au problème d'étiquetage. Cependant, il est probable qu'aucune des séries ne donne la solution idéale pour toutes les étiquettes. Comme il est mentionné au début du chapitre, trouver un modèle qui va permettre un bon clustering pour toutes les étiquettes est un problème sans doute impossible. D'un autre coté, il est plus facile de trouver un modèle qui donnera des résultats adéquats pour une seule étiquette.

Nous avons mentionné à la section 2.4.2 qu'une étiquette pouvait se retrouver dans des clusters passablement différents dépendant des variations dans l'objet étiqueté. Il est donc possible qu'un seul modèle ne fasse pas l'affaire pour une étiquette particulière. Pour cette raison, la sélection ne se fait pas directement au niveau modèle, mais plutôt au niveau des clusters générés par les modèles.

Un petit exemple va aider à éclaircir ce dernier point. Supposons que nous ayons 9 imagettes ayant la même étiquette. Pour simplifier nous n'indiquerons pas les imagettes qui n'ont pas encore reçu d'étiquettes. Nous avons aussi deux modèles, a et b, qui donnent les clusters suivants avec l'étiquette en question :

Modèle a : {1, 2, 3, 4} {5, 6} {7, 8} {9}

Modèle b : {1, 2} {3, 4} {5, 6, 7, 8} {9}

Résultat : {1, 2, 3, 4} {5, 6, 7, 8} {9}.

Si nous utilisons les mêmes critères que lors de la création des clusters (Figure 2.9), il est

facile de calculer le résultat de la sélection. En d'autres mots, nous utilisons le meilleur de chaque modèle.

Algorithme de sélection globale

Nous utilisons présentement une approche exhaustive pour la sélection des clusters. Cette approche exhaustive est combinée avec des heuristiques pour réduire le champs de recherche. Il est aussi possible d'utiliser des réseaux de neurones pour cette étape. Minka nous en donne un exemple dans [20] en utilisant un réseau-auto-organisé.

1. Regrouper tous les clusters de tous les modèles qui ont la même étiquette.
2. Faire un tri des clusters en ordre décroissant du nombre d'éléments qui ont une étiquette.
3. Enlever les clusters redondants et les singletons (heuristique 1 et 2 de la fig. 2.14).
4. Faire une liste des imagettes qui ont une étiquette.
5. Comparer, dans l'ordre, toutes les combinaisons de clusters avec la liste précédente tout en utilisant les heuristiques 3 et 4 de la Figure 2.14. Si toutes les imagettes de la liste de l'étape 4 se retrouvent dans la combinaison nous avons la solution et l'algorithme s'arrête.

Figure 2.13: Algorithme de sélection globale.

L'algorithme présenté à la Figure 2.13 va nous donner en sortie la liste de clusters qui satisfait à nos critères. Si nous utilisons les clusters de l'exemple précédent, nous pouvons facilement suivre les étapes énumérées.

Étape 1 : {1, 2, 3, 4}, {5, 6}, {7, 8}, {9}, {1, 2}, {3, 4}, {5, 6, 7, 8}, {9}

Étape 2 : {1, 2, 3, 4}, {5, 6, 7, 8}, {1, 2}, {3, 4}, {5, 6}, {7, 8}, {9}, {9}

Étape 3 : {1, 2, 3, 4}, {5, 6, 7, 8}, {1, 2}, {3, 4}, {5, 6}, {7, 8}

Étape 4 : {1, 2, 3, 4, 5, 6, 7, 8}

Étape 5 : {1, 2, 3, 4}, {5, 6, 7, 8}

L'étape 5 est la seule qui demande un peu d'explication. Les clusters sont premièrement pris un à la fois, en commençant par le premier, et comparés à la liste obtenue à l'étape 4. Comme cela ne fonctionne pas ici les clusters sont pris deux à la fois comme suit : 1-2, 1-3, 1-4, ... , 2-3, 2-4, ... Les combinaisons sont additionnées ensemble. Ainsi {1, 2, 3, 4} + {5, 6} = {1, 2, 3, 4, 5, 6}, où + dénote l'union ensembliste. Si cela ne fonctionne pas les clusters sont pris trois à la fois, mais toujours dans l'ordre précis 1-2-3, 1-2-4, ... Nous sommes certains qu'il y a une solution car les imagettes de l'étape 4 sont prises à partir de la liste que nous utilisons pour les combinaisons.

1. Éliminer les clusters qui se répètent.
2. Éliminer les singletons (clusters qui ne sont composés que d'une seule imagette).
3. Arrêter l'étape quand le nombre d'éléments qui reste est trop petit pour remplir la liste.
4. Arrêter l'algorithme quand le nombre de clusters dans la solution devient trop grand.

Figure 2.14: Heuristiques.

La Figure 2.14 nous donne la liste des heuristiques découvertes jusqu'ici. Les deux premières règles enlevent les clusters inutiles **avant** que le processus de comparaisons

combinatoires soit amorcé. La règle trois arrête les comparaisons combinatoires quand cela devient inutile. Cette règle est possible car les clusters sont triés par ordre décroissant du nombre d'éléments. Finalement, la dernière règle arrête le processus au complet quand l'amorce de solution indique qu'il peut ne pas y avoir de bon clustering pour cette étiquette. Dans ce dernier cas, il n'y aura pas d'étiquetage.

2.6 Conclusion

Dans ce chapitre nous avons décrit en détail le fonctionnement du système d'étiquetage automatique de l'image. Nous avons aussi vu le pourquoi et le comment de chacun des éléments du système sauf pour le comment de la section clustering qui sera élaboré en détail dans le chapitre trois.

Le pourquoi occupe une place importante dans ce chapitre car nous croyons qu'il est nécessaire de justifier les choix de méthodes et d'algorithmes. Nous avons expliqué pourquoi nous avons choisi de créer un système qui fonctionne avec un ensemble de modèles, pourquoi nous avons utilisé un clustering en passant par des arbres hiérarchiques et pourquoi nous avons décidé de créer notre propre algorithme de clustering.

Le comment doit, par nécessité, entrer dans les détails techniques et en plus des descriptions nous avons élaboré sur certains algorithmes importants et avons même donné quelques exemples pertinents. L'aspect pratique et les exemples de l'utilisation du système se retrouvent au chapitre 4.

CHAPITRE 3

Algorithmes de clustering

Comme il est souligné dans l'introduction, nous soumettons dans ce projet une nouvelle méthode de clustering. Nous avons donné le nom d'**arbre des distances** à notre nouvelle méthode. C'est une méthode agglomérative géométrique, mais jusqu'ici nous n'avons rien trouvé d'identique dans la littérature. Le clustering est utilisé dans plusieurs domaines (astronomie, biologie, chimie, démographie, économie, ...) et la littérature sur le sujet se retrouve donc dans les publications de chaque domaine. Beaucoup de livres en parlent, mais il y a très peu de livres qui se concentrent uniquement sur ce sujet. Nous avons soumis notre algorithme à plusieurs tests pour savoir s'il était capable de faire un bon clustering et s'il n'y avait pas de problèmes importants. Nous espérons que notre nouvel algorithme soit une contribution dans le domaine.

Dans la première section de ce chapitre nous élaborons quelque peu sur le sujet de clustering afin de mieux comprendre le sujet et de situer **l'arbre des distances** par rapport

aux autres algorithmes. Nous expliquons ensuite à fond le fonctionnement du nouvel algorithme et décrivons les tests qui lui ont été soumis. L'algorithme des voisins-communs de Jarvis-Patrick est utilisé pour comparaison.

3.1 Généralités sur le clustering

Comme il a été mentionné au chapitre deux, l'opération de clustering est équivalente à séparer un ensemble en sous-ensembles disjoints de telle sorte que tous les éléments d'un sous-ensemble jouissent d'une propriété commune. Même pour un petit nombre d'éléments le nombre de possibilités est très grand. Par exemple, pour seulement 25 éléments il y a plus de 4×10^{18} groupements possibles [1]. Un algorithme de clustering ne saurait donc être parfait et l'on demande seulement qu'il fournisse une solution acceptable dans un temps minimal.

Nous demandons aux algorithmes de clustering de nous *révéler* la structure qui se trouve dans les données que nous lui soumettons. Il faut faire attention que l'algorithme n'*impose* pas une structure qui n'existe pas. En fait, ce concept de structure se retrouve dans l'algorithme lui-même et donc, l'algorithme ne peut découvrir que les clusters qui adhèrent à cette structure.

Une série de clusters n'est pas une fin en soi, mais seulement une partie d'un processus qui nécessite une phase de clustering. Il n'est donc pas nécessaire de trouver le meilleur algorithme de clustering qui soit si cela ne fait que peu de différence à la solution finale du problème qui nous préoccupe.

Dans le reste de cette section, nous allons examiner les questions qu'il faut se poser lors de la sélection d'un algorithme de clustering.

3.1.1 Comparaison des éléments d'un cluster

Nous venons de mentionner que les éléments d'un cluster jouissent de propriétés en commun. Cette phrase est quelque peu vague, mais nous devons cependant lui donner une définition mathématique si nous voulons l'incorporer dans un algorithme. En d'autre terme, il faut pouvoir comparer les éléments entre eux et pouvoir dire que certains éléments sont plus semblables entre eux que d'autres. Deux éléments sont semblables (à un certain point de vue) s'ils ont des caractéristiques en commun.

Un élément possède, ou non, une caractéristique ce qui nous donne déjà une mesure de zéro ou un. Des mesures plus détaillées sont souvent possibles. La solution généralement utilisée, est de définir une mesure qui soit représentable dans un espace euclidien. Ainsi, les éléments peuvent être représentés par des points dans un espace vectoriel ce qui nous permet de calculer la distance entre les points. Ceci nous donne une vue géométrique des éléments d'un ensemble et deux éléments similaires devraient être près l'un de l'autre dans l'espace vectoriel.

3.1.2 Types de clusters

Comme la plupart des méthodes de clustering utilisent une approche géométrique, il est possible de visualiser ceux-ci. La Figure 3.1 nous donne une idée des types de clusters

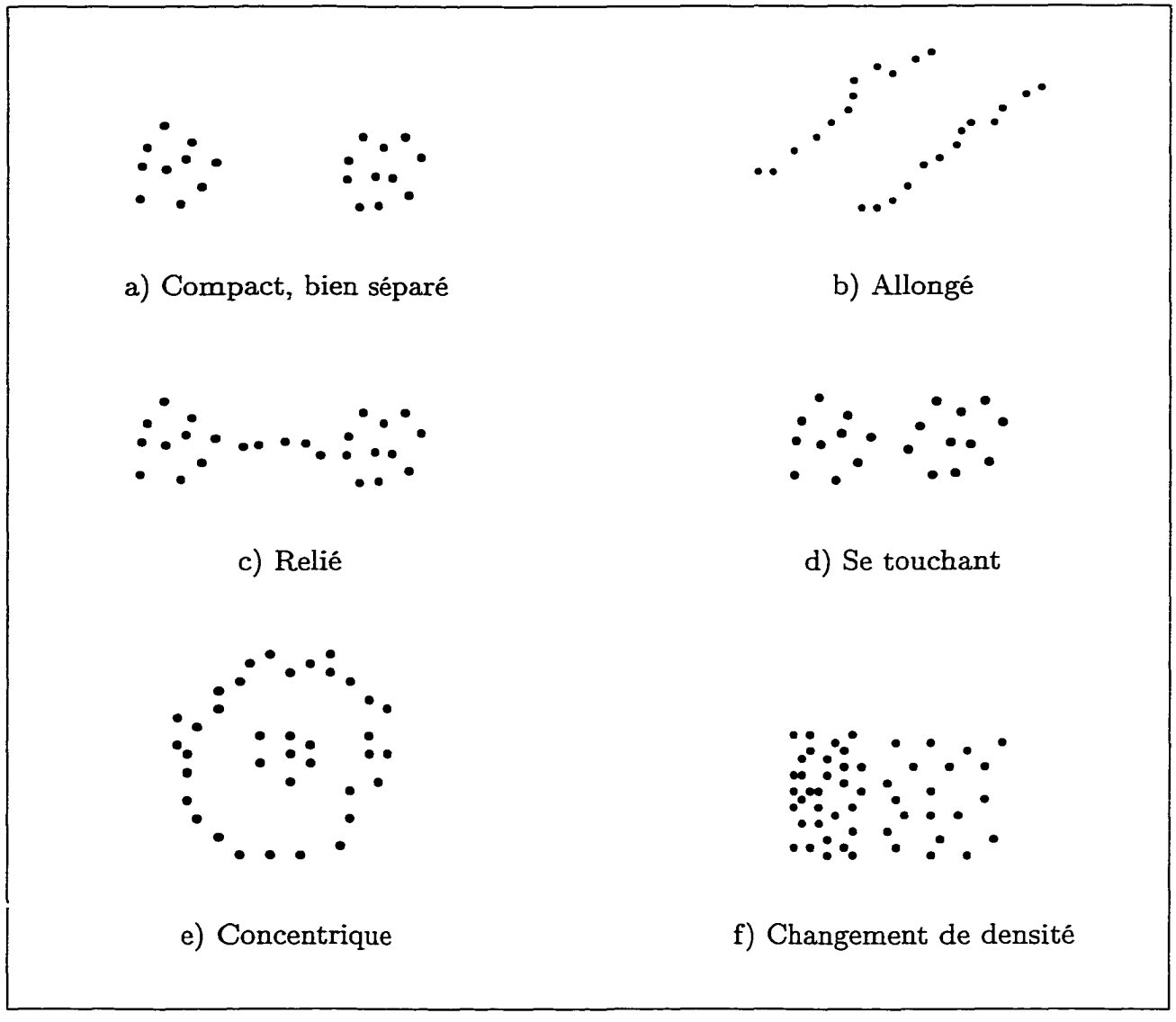


Figure 3.1: Différents types de clusters.

qu'il est possible de rencontrer. Dans le cluster idéal tous les éléments occuperaient le même point. A défault de cet ideal, les clusters de forme sphérique (cluster globulaire) sont les plus faciles à identifiés. Toutes les méthodes de clustering devraient donner de bons résultats face à ce genre de clusters [15].

En fait, plusieurs méthodes de clustering utilisent l'idée même d'un point central à chaque cluster. Pour savoir à quel cluster un nouveau point appartient il suffit de calculer la distance entre ce point et tous les points centraux et de choisir le plus court. La méthode des K-moyennes [7] peut être utilisée pour calculer ces points centraux si on connait le nombre de clusters.

Les méthodes à clusters globulaires peuvent avoir des problèmes avec des clusters allongés (Figure 3.1 b). Par exemple, les types de clusters illustrés en e) et f) vont certainement poser des problèmes aux approches globulaires. Une correction au système de coordonnées ou une normalisation des données peut souvent solutionner ce problème. Les méthodes de clustering par voisins-communs peuvent être utilisées pour solutionner ce genre de problèmes car elles sont insensibles à la forme géométrique des clusters.

3.1.3 Méthodes de clustering

Il existe deux classes principales dans les méthodes de clustering: les méthodes hiérarchiques et non-hiérarchiques (voir Figure 3.2). Les méthodes hiérarchiques demandent généralement beaucoup plus de ressources car elles doivent créer une hiérarchie de partition plutôt qu'une seule. Elles sont cependant plus riches en informations car elles indiquent comment de petits clusters peuvent être regroupés en clusters plus grands.

Les méthodes agglomératives fonctionnent de bas en haut en agglomérant les éléments en petits clusters puis en fusionant ces clusters pour former des clusters de plus en plus gros. Les mèthodes par division fonctionnent de haut en bas en divisant les données en sous-groupes puis en redivisant progressivement les sous-groupes, etc.

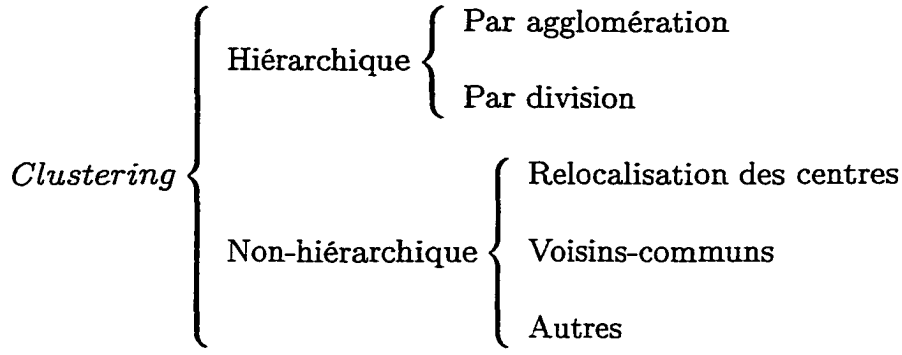


Figure 3.2: Classification des méthodes de clustering.

Comparativement aux méthodes hiérarchiques, les méthodes non-hiérarchiques donnent comme résultats une série de clusters bien définis. Certaines de ces méthodes demandent même le nombre de clusters comme paramètre, mais d'autres peuvent ajouter ou retrancher des clusters au besoin.

Parmi les méthodes non-hiérarchiques mentionnons les méthodes de relocalisation des centres qui sont très populaires car facile à comprendre et à utiliser. La méthode des K-moyennes, aussi appelée ISODATA [7] [15], est un exemple d'une méthode qui fonctionne par approximation en recalculant les centres des clusters selon les points environnants.

Les méthodes par voisins-commun n'utilisent pas de point central, mais regroupent ensemble les points qui sont près l'un de l'autre ou similaires d'une certaine façon. La méthode de Jarvis-Patrick est la plus connue des méthodes de voisins-communs.

3.2 Méthode de Jarvis-Patrick

Nous avons inclus, dans notre travail, la méthode de Jarvis-Patrick pour deux raisons principales: c'est une méthode bien connue qui a fait ses preuves et c'est la méthode utilisée par l'équipe du MIT.

```
variable  N              // nombre d'élements
          K              // nombre de voisins
          lvc[N][N]      // liste des voisins-communs
          vv[N]          // vecteur de voisins

Calcul_lvc( )
      pour i = 1 à N
            et pour j = 1 à N
                  lvc[i][j].np = j // numéro du point
                  lvc[i][j].dist = Distance( Point[i], Point[j] )
            Tri( lvc[i] )
fin
```

Figure 3.3: Algorithme de Jarvis-Patrick, partie A.

Cette méthode utilise deux paramètres: K, le nombre de voisins à considérer et K_{min} le nombre de voisins que deux points doivent avoir en commun parmi leur K voisins respectifs. Selon la version standard de cette méthode, le clustering se fait en deux étapes. Dans la première étape, les K voisins les plus près de chaque objet sont identifiés. Dans la deuxième étape, deux objets i et j se retrouvent dans le même cluster s'ils obéissent aux conditions suivantes:

```
Test_voisin( a, b, k )
        pour i = 1 à k si lvc[b][i] == a retourne vrai
        retourne faux
fin
Voisin_commun( a, b, k )
        si Test_voisin( a, b, k ) != vrai retourne faux
        si Test_voisin( b, a, k ) != vrai retourne faux
        pour i = 1 à k
                si Test_voisin( lvc[a][i], b, k ) == vrai nv++
        si nv >= K_min retourne vrai
        retourne faux
fin
Regroupe_voisin( a, b )
        ln = min( vv[a], vv[b] )
        hn = max( vv[a], vv[b] )
        pour i = 1 à N si vv[i] == hn alors vv[i] = ln
fin
Algorithme_JP( k )
        pour a = 1 à N
                et pour b = a+1 à N
                        si Voisin_commun( a, b, k ) alors Regroupe_voisin( a, b )
fin
```

Figure 3.4: Algorithme de Jarvis-Patrick, partie B.

- i est un K voisin de j
- j est un K voisin de i
- i et j ont au moins K_{min} de leur K voisin en commun.

Le problème principal se situe dans le choix judicieux des paramètres. Si K est trop petit, nous nous retrouvons avec plusieurs petits clusters sans grandes significations. Si K est trop grand les clusters seront regroupés ensemble.

Les algorithmes pour les deux étapes se retrouvent aux Figures 3.3 et 3.4. Dans la partie A, la distance entre tous les points est calculée et un tri fait en sorte que tous les voisins, pour chaque point, se retrouvent en ordre de distance. La partie B nous donne comme résultat un vecteur de voisins. Dans ce vecteur, tous les points qui sont voisins l'un de l'autre possèdent le numéro du voisin le plus petit. Ainsi, une lecture du vecteur nous donne tous les clusters.

Cet algorithme est une méthode de clustering non-hiérarchique, mais il est quand même possible de construire un arbre de clusters en exécutant l'algorithme itérativement avec des K de plus en plus grand. L'algorithme de Jarvis-Patrick se prête bien à cet exercice car la partie qui prend le plus de temps calcul, la partie A, n'a besoin d'être calculée qu'une fois. Les itérations se font seulement avec la partie B.

3.3 Arbre des distances

L'arbre des distances pourrait être classé comme une méthode de clustering hiérarchique par agglomération. C'est aussi une méthode à clusters globulaires ce qui implique que l'algorithme peut avoir des problèmes avec des clusters non sphériques. Étant donné une liste de points dans un espace vectoriel, l'algorithme (voir Figure 3.6) va créer un arbre

hiérarchique binaire en fonction de la distance entre les points de l'espace vectoriel.

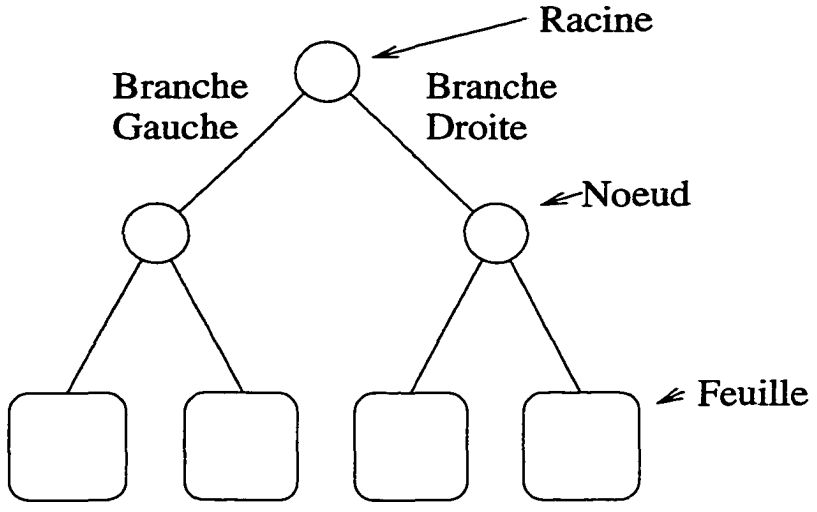


Figure 3.5: Parties de l'arbre hiérarchique.

La figure 3.5 nous montre les parties de l'arbre. Il faut noter que la racine, et les feuilles, sont aussi des noeuds. Chaque noeud de l'arbre contient les informations suivantes: un point dans l'espace vectoriel, un rayon, un poids, et possiblement une étiquette. Les noeuds terminaux (les feuilles) représentent les points d'entrée (les imagettes). Une feuille a un poids de 1, un rayon de 0, et peut avoir une étiquette. Les noeuds non terminaux représentent des sphères regroupant les feuilles. Le poids de ces noeuds est égal au nombre de feuilles qu'il englobe. Dans un sens, ces points et rayons représentent des voisinages sphériques, (voir Figure 3.7) les points sont les centres des sphères et les rayons sont évidemment les rayons des sphères. La détermination de ces voisinages est une partie importante de l'algorithme. Le calcul optimal du voisinage reste un sujet de recherche.

La construction de l'arbre se fait une feuille à la fois. L'algorithme a comme données d'entrée la racine d'un arbre et une feuille. À la sortie de l'algorithme nous retrouvons un

```
Arbre_distance( branche, feuille )
        si branche.poids == 0 retourne feuille
        si branche.poids == 1 retourne Nouveau_noeud( branche, feuille )
        si Distance( branche, feuille ) > branche.rayon alors
                retourne Nouveau_noeud( branche, feuille )
        si Distance( branche.gauche, feuille ) < Distance( branche.droite, feuille )
                alors branche.gauche = Arbre_distance( branche.gauche, feuille )
                sinon branche.droite = Arbre_distance( branche.droite, feuille )
        branche.centre = Nouveau_noeud( branche.gauche, branche.droite )
        branche.rayon = max( Distance( branche, branche.gauche ),
                             Distance( branche, branche.droite ) )
        retourne branche
fin
```

Figure 3.6: Algorithme de construction de l'arbre des distances. Les fonctions **Distance** et **Nouveau_noeud** sont expliquées dans le texte.

nouvel arbre qui contient la feuille ajoutée à l'arbre d'entrée. C'est un algorithme récursif et nous n'avons que trois cas à traiter: l'arbre est vide, l'arbre ne contient qu'une feuille, l'arbre contient un noeud (non terminal). Si l'arbre contient un noeud, les distances entre la feuille et les deux branches du noeud sont calculées. La feuille est ajoutée à la branche qui est la plus près de la feuille par un appel récursif de l'algorithme.

L'algorithme a besoin de deux fonctions, l'une pour calculer la distance entre deux noeuds ou feuilles, et l'autre générer un nouveau noeud:

- **Distance(a, b) :** Calculer la distance d entre deux noeuds a et b selon la formule

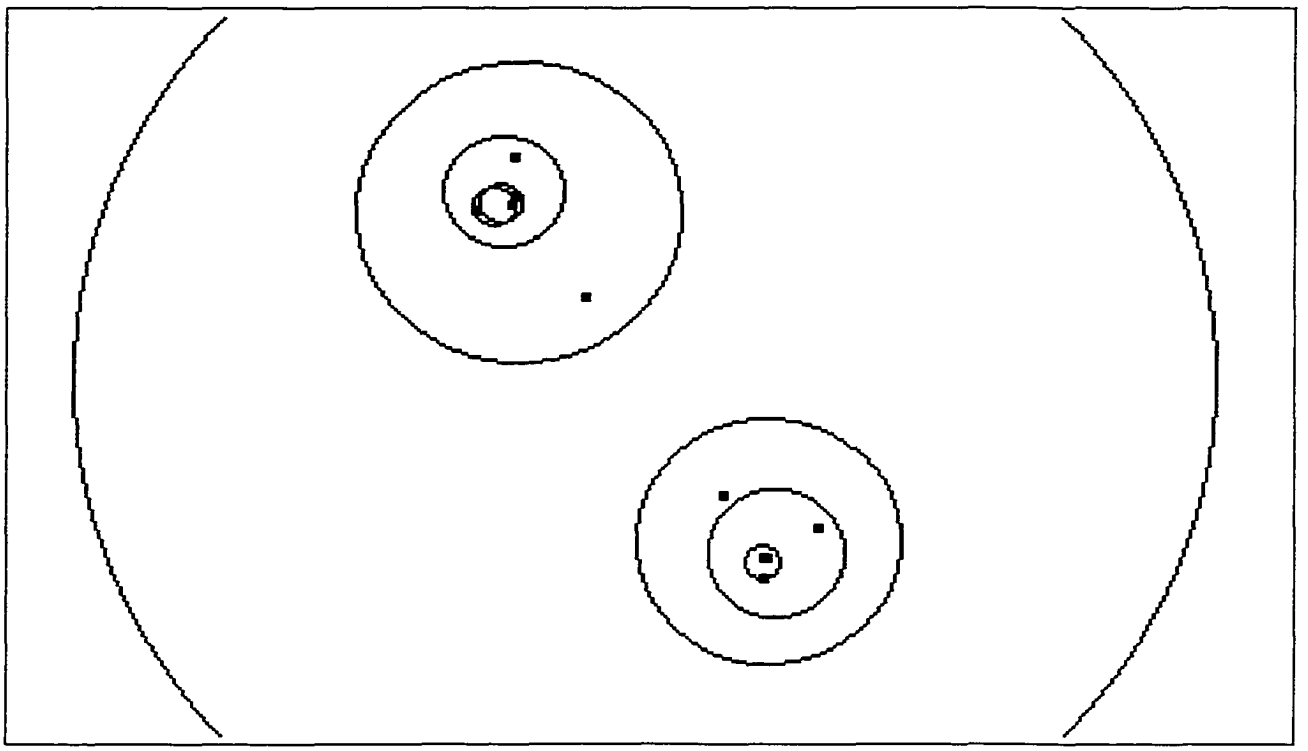


Figure 3.7: Noeuds et voisinages.

usuelle:

$$d = \sqrt{\sum_i (a_i - b_i)^2}$$

- **Nouveau_noeud(a, b) :** Calculer un nouveau noeud c qui est sur la droite qui passe par les deux noeuds donnés. La position de ce nouveau noeud sur la ligne dépend d'un poids qui est attaché à chaque noeud. Par exemple, si le poids P_a est égal à 1 et le poids $P_b = 2$, le nouveau noeud sera au 2/3 sur la ligne entre a et b.

$$c = \frac{aP_a + bP_b}{P_a + P_b}$$

L'algorithme est exécuté pour chaque feuille selon les étapes suivantes:

- Si la racine est vide: retourner la feuille comme solution.

- Si la racine est une feuille: créer un noeud et y ajouter les deux feuilles, le poids du noeud sera de 2. Le nouveau noeud sera au milieu entre les deux feuilles et le rayon sera tel que les deux feuilles seront à l'intérieur du nouveau voisinage.

- Si la racine est un noeud: vérifier si la nouvelle feuille est à l'intérieur ou à l'extérieur du voisinage de ce noeud.

  - Si la nouvelle feuille est à l'extérieur du voisinage: créer un nouveau noeud et y ajouter la feuille et le noeud de la racine. Calculer le nouveau point du centre, le rayon et le poids.

  - Si la nouvelle feuille est à l'intérieur du voisinage: comparer la distance entre la feuille et les deux sous-noeuds. Choisir le sous-noeud qui est à la moindre distance et faire un appel récursif de l'algorithme avec le sous-noeud comme racine. A la suite de l'appel, recalculer les informations du noeud basé sur les nouvelles valeurs du sous-noeud que l'appel a retourné.

Comme l'arbre des distances est un arbre binaire, le nombre de noeuds est égal au nombre de feuilles moins un. Le vecteur de caractéristiques est le seul élément possiblement encombrant de ces noeuds et feuilles. L'espace utilisé par l'arbre est donc modeste surtout si comparé aux méthodes où il faut calculer une matrice de distance entre tous les points. L'arbre peut donc facilement être stocké sur disque pour être réutilisé lors de l'ajout de nouveaux points.

3.4 Clusters artificiels

Dans le but de vérifier si notre méthode de clustering donne de bons résultats, des clusters artificiels de formes sphériques sont générés dans un espace à deux dimensions. Nous pouvons spécifier, pour chaque test, la configuration des clusters ainsi que le nombre de points dans chaque cluster. Si les clusters artificiels sont bien séparés, les résultats devraient être parfaits. Quand les clusters artificiels sont proches l'un de l'autre, il est possible qu'un point du cluster a se retrouve dans le cluster b.

Dans le but d'obtenir des clusters sphériques qui suivent une loi normale nous utilisons des coordonnées polaires pour calculer les points d'un cluster puis nous convertissons les points en coordonnées cartésiennes pour les besoins de l'algorithme de clustering. Comme nous utilisons ici un espace à deux dimensions nous devons générer deux nombres pour spécifier la position du point. Dans un système de coordonnées polaires, ces deux nombres sont l'angle et le rayon. Le premier nombre généré, l'angle, doit suivre une loi uniforme et la valeur du nombre doit être entre 0 et 2π. Le deuxième nombre, le rayon, suit la loi normale N(0,1) dont la courbe est:

$$F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt.$$

La méthode utilisée pour la génération des nombres suivant la loi normale est décrite dans Knuth [18] Vol.2, p104.

La Figure 3.8 donne les configurations des clusters qui furent utilisées pour vérifier la nouvelle méthode. Nous commençons par deux clusters bien séparés. Cette configura-

tion ne devrait donner aucun problème à n'importe quel algorithme de clustering. Les problèmes apparaissent quand les clusters se touchent.

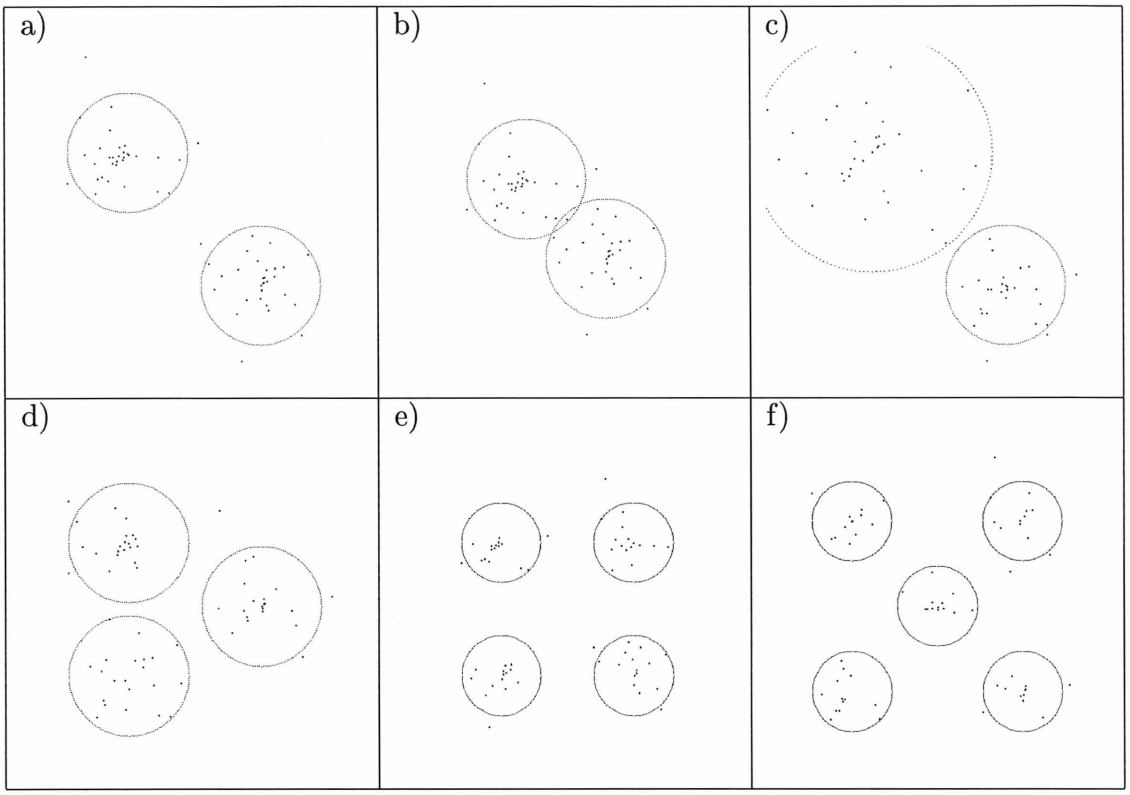


Figure 3.8: Configurations de clusters.

3.5 Résultats

Nous avons utilisé les configurations de clusters artificiels décrites à la section précédente pour vérifier la précision du clustering et la vitesse d'exécution pour la méthode de Jarvis-Patrick et la méthode de l'arbre des distances. Chaque configuration contient exactement 64 points pour que toutes les configurations puissent être comparées entre elles.

3.5.1 Précision du clustering

Les tests préliminaires, faits au début du projet avec quelques clusters différents, semblaient indiquer que les deux méthodes donnaient le même genre de précision. Si les clusters étaient bien séparés, les deux méthodes donnaient le bon résultat. Si les clusters étaient très près l'un de l'autre, les deux faisaient le même type d'erreur. Ces erreurs étaient dues au fait que certains points pouvaient aussi bien être classés dans un cluster que dans l'autre.

Configuration	Jarvis-Patrick	Arbre des distances
a	0	0
b	2	3
c	3	2
d	2	1
e	0	0
f	0	1

Tableau 3.1: Nombre d'erreurs dans les méthodes évaluées.

Le Tableau 3.1 indique le nombre d'erreurs pour les deux méthodes étudiées avec les configurations de clusters de la Figure 3.8. Mentionnons que les configurations b, c et d possèdent chacunes quelques points dont le classement est incertain.

Ces derniers tests indiquent que l'arbre des distances fonctionne bien quand il y a peu de clusters ou quand les clusters sont séparés par de grandes distances relativement à la grosseur des clusters. La méthode de Jarvis-Partick donne le même genre d'erreur dans les cas simples, mais fonctionne mieux quand il y a beaucoup de clusters.

3.5.2 Vitesse d'exécution

La complexité de la méthode de Jarvis-Patrick se situe entre $O(N^2)$ et $O(N^3)$, dépendant de l'implémentation. La complexité de la méthode de l'arbre des distances est $O(N \log_2(N))$. Les valeurs du Tableau 3.2 ont été calculées à partir d'un Pentium de 150 MHz qui utilise Linux comme système d'exploitation.

Points	Jarvis-Patrick	Arbre des distances
64	0.27	0.04
128	1.74	0.08
256	15.95	0.14
512	134.83	0.25

Tableau 3.2: Temps de l'UCT en seconde pour chaque méthode.

L'arbre des distances est certainement plus rapide que la méthode de Jarvis-Patrick, mais son avantage principal provient du fait que de nouveaux points peuvent être ajoutés à un arbre existant. L'ajout de 64 nouveaux points à un arbre qui en contient déjà 512 ne prend que 0.06 secondes. Pour ce genre d'opération, la méthode de Jarvis-Patrick prend environ 150 secondes car il faut refaire tous les calculs.

3.6 Conclusion

L'idée de l'arbre des distances nous est apparue durant la recherche d'un algorithme plus performant que celui de Jarvis-Patrick. Comme l'implémentation n'en était pas trop difficile, l'algorithme fut programmé. Les tests préliminaires semblaient prometteurs et la décision fut prise d'utiliser cet algorithme.

Il est difficile de savoir s'il aurait été plus profitable de continuer la recherche pour un algorithme déjà existant. Nous avons certainement mis beaucoup de temps pour augmenter l'efficacité de l'arbre des distances, principalement en "jouant" avec les paramètres.

Une avenue possible pour améliorer la précision de l'arbre des distances est d'en faire un arbre n-aire plutôt qu'un arbre binaire. L'implémentation en serait certainement plus difficile et il n'est pas sûr que les avantages sur l'utilisation de la mémoire et sur la vitesse soient préservés.

CHAPITRE 4

Description du système

Ce chapitre documente la partie pratique de ce mémoire. La première section se veut un manuel de l'utilisateur pour le système d'étiquetage d'images. Les outils utilisés pour la vérification des programmes sont décrits dans la deuxième section. Les détails d'implémentation qui peuvent ête utiles à ceux qui veulent poursuivre ce travail se retrouvent dans la troisième section. Finalement, nous finissons avec les résultats du projet.

4.1 Manuel de l'utilisateur

SELIM[1] a été conçu dans le but de faire l'étiquetage d'un nombre illimité d'images. L'utilisation de **SELIM** se divise en deux phases distinctes. Dans la première phase l'usager instruit le système à l'aide d'une série d'images typiques. C'est la phase d'ap-

[1] **S**ystème pour l'**E**tiquetage d'une **L**iste d' **IM**ages.

prentissage. Dans la deuxième phase, le système peut être utilisé pour faire l'étiquetage d'un nombre illimité d'images. Le résultat sera, pour chaque image, un fichier contenant les étiquettes que le système propose pour l'image. Ces fichiers d'étiquettes pourront ensuite être utilisés par un autre système.

Le système d'étiquetage d'image a été conçu en regard de l'utilisateur final et utilise une interface graphique basée sur Amulet[2]. Quelques-uns des critères suivis lors de la conception du système sont la facilité d'utilisation, l'apparence professionnelle et une bonne performance.

Le système peut être utilisé en mode interactif ou en traitement par lots. En mode interactif, l'usager peut importer une nouvelle image, y ajouter des étiquettes et voir les résultats de l'étiquetage automatique sur le champ. L'ajout des images et des étiquettes se fait avec la souris. Le traitement par lots n'utilise pas l'interface graphique et sert à traiter de vaste quantités d'images automatiquement. Enfin, la base de données de **SELIM** peut être convertie en fichier texte ou vice-versa. Ceci est utile pour recréer la base de données sans avoir à refaire toutes les entrées des données.

4.1.1 Préparatifs

Au début d'un nouveau projet d'étiquetage, l'usager devrait créer un répertoire pour la base de données du projet et un autre pour les images elles-mêmes. En effet, il est plus facile pour la gestion du projet de garder les fichiers images et les fichiers **SELIM** séparés.

[2]Voir section implémentation

Pour débuter le programme en mode interactif, l'usager se positionne dans le répertoire de la base et tape la commande `selim` (voir Figure 4.1).

Si la base de données n'existe pas, le programme va en créer une automatiquement. Comme les images peuvent résider dans un répertoire différent, la première étape devrait être la spécification du répertoire des images avec l'option menu `Fichier+Répertoire`. Le nom du répertoire va être sauvegardé dans la base de données.

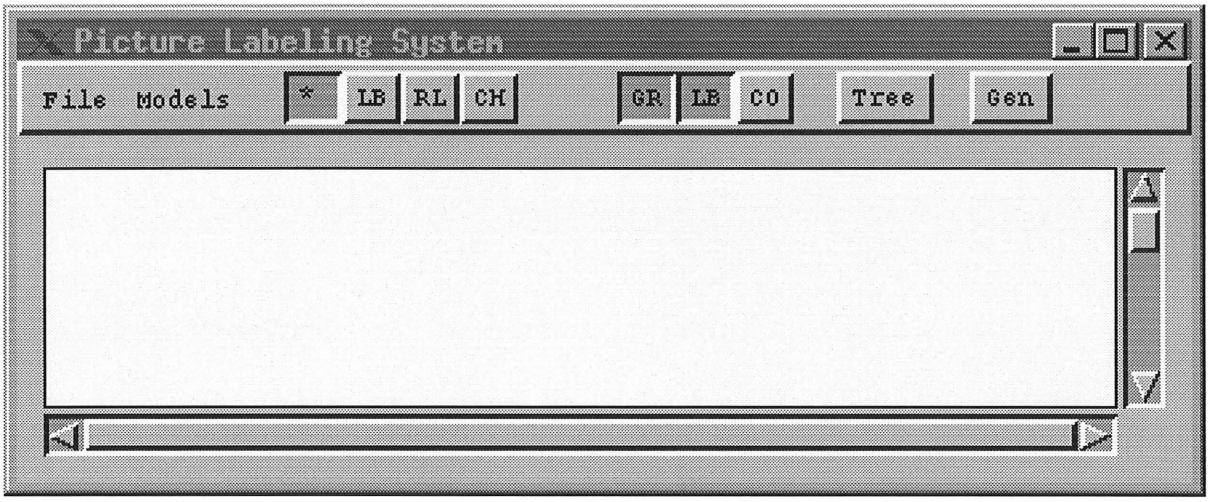


Figure 4.1: Programme **selim** à l'ouverture.

4.1.2 Ajout d'images

Pour ajouter une nouvelle image dans la base il faut utiliser `File+New`. Dans la version présente du programme, l'image doit être en format **gif**[3] et aussi en format **ppm**. Dans la boite dialogue il faut spécifier le fichier gif.

Les informations accumulées sur l'image sont sauvegardées dans la base sous le nom de

[3]Problème d'interface, voir section 4.3.1

l'image moins son extension. Pour reprendre une image déjà stockée, utilisez File+Open. Présentement, la liste des images dans la base n'est pas affichée. La commande séparée *piclist* peut être utilisée pour afficher la liste.

Toutes les informations ajoutées au système sont sauvegardées automatiquement. Il n'y a pas de commande File+Save. Ceci est dû au fait qu'il est difficile d'enlever les données une fois qu'elles sont incorporées dans les arbres. Donc, une image ne peut être directement enlevée de la base. Toutefois, il est possible de traduire la base en un fichier texte, d'enlever l'image et de recréer la base.

4.1.3 Ajout d'étiquettes

Pour étiqueter l'image, un quadrillage est superposé sur l'image pour créer des imagettes. Les trois boutons suivant servent à contrôler la grille, les étiquettes et un système de coordonnées. Voir Figure 4.2 pour l'apparence du programme à ce stade.

GR Montrer grille.

LB Montrer étiquettes.

CO Montrer coordonnées.

Les 4 boutons qui suivent définissent les actions sur les imagettes. Ces boutons ne peuvent être actionnés qu'un seul à la fois.

** Ne rien faire.

Figure 4.2: Programme **selim** avec image.

LB Ajout d'étiquettes.

RL Retrait d'étiquettes.

CH Montrer l'histogramme des couleurs.

L'usager doit premièrement sélectionner l'action à accomplir puis utiliser la souris pour spécifier la ou les imagettes concernées. Pour ajouter ou enlever des étiquettes, un rectangle peut spécifier un nombre d'imagettes en cliquetant sur la case de départ et relâchant le bouton sur la case d'arrivée.

Lors de l'ajout d'étiquettes, le bouton du milieu de la souris sert à prendre une copie d'une étiquette et le bouton de droite sert à copier l'étiquette sur une sélection d'imagettes.

Le programme permet de visualiser l'histogramme des couleurs d'une imagette ou même de deux imagettes prises ensemble dans le but de les comparer (voir Figure 2.5 pour un exemple). Pour ce faire, l'usager doit cliqueter deux fois sur la même imagette ou sur deux imagettes différentes.

4.1.4 Génération automatique d'étiquettes

Pour générer de nouvelles étiquettes, le système doit répéter les étapes 1 à 3 pour chaque modèle et ensuite terminer avec l'étape 4.

1. Calculer le vecteur du modèle pour chaque imagette.
2. Ajouter chaque vecteur dans l'arbre des distances.
3. Parcourir l'arbre des distances pour en extraire des clusters qui ne contiennent qu'un seul type d'étiquette.
4. Combiner toutes les propositions de l'étape 3 pour trouver la meilleure solution. Apposer de nouvelles étiquettes là où c'est nécessaire.

Faire le calcul de l'étape 1 et 2 pour toutes les imagettes du système prendrait beaucoup de temps. Pour rendre le système utilisable, le résultat de ces calculs est sauvegardé dans plusieurs fichiers, un par modèle. Chaque image ajoutée au système est ajoutée automatiquement à chaque fichier modèle.

Les étapes 3 et 4 sont relativement rapides. L'usager peut manuellement changer quelques étiquettes et voir les étiquettes générées en cliquetant le bouton **Gen**.

Par défaut, le système va trouver la meilleure *couverture* en combinant tous les modèles. Il est aussi possible d'utiliser un seul des modèles dans la liste ci-dessous. Cette liste va grandir à mesure que de nouveaux modèles s'ajouteront au système. La description des modèles de cette liste se retrouvent au chapitre deux dans la section 2.3.

- All (Utilise tous les modèles)
- Color Histogram
- Hue histogram
- Mean Color
- Co-occurence 1
- Co-occurence 2

Enfin, les deux derniers boutons sont utilisés pour générer les étiquettes et afficher l'arbre des distances.

Gen Générer les étiquettes en utilisant le modèle sélectionné.

Tree Afficher l'arbre des distances.

4.1.5 Traitement par lots

Usage:[4] selbatch [-o répertoire] [-d répertoire] [-l fichier] image ...

Options:

-o Nom du répertoire pour les fichiers résultats.
(par défaut, c'est le répertoire de l'image)

-d Nom du répertoire à traiter.

-l Fichier qui contient une liste d'images à traiter.

C'est la version traitement par lots du système. Chaque image va être étiquetée et les résultats sauvegardés dans un fichier texte qui porte le nom de l'image.

4.1.6 Conversions

Usage: seltxt {-r,-w} file

Options:

-r Lecture du fichier.

-w Écriture du fichier.

Ce programme est utilisé pour traduire la base de données en un fichier texte ou de recréer la base à l'aide du fichier texte.

[4]Nous utilisons la notation Unix standard: [ ] sont pour les items optionnels et { } pour les choix.

Format du fichier texte:

```
DIR: ../pic/
GIF: mount.gif
LABELS:
0, 0: sky
1, 2: grass
END
```

Les sections `GIF`, `LABELS`, ... , `END` sont répétées pour chaque image.

4.1.7 Programmes connexes

Les programmes qui suivent font partie intégrante du système. Chacun d'eux accomplit une tâche bien spécifique. Ils sont généralement invoqués automatiquement par le programme principal, mais ils peuvent être utilisés directement par l'utilisateur.

- **Création de l'arbre**

Usage: covers {all, number} [-r]

Options:

nombre	Génère les étiquettes pour un arbre. Le nombre fait référence au numéro du modèle.
all	Génère toutes les étiquettes pour tous les arbres.
-r	Ne pas construire l'arbre des distances, le lire dans le fichier.

C'est le programme qui accomplit les étapes 1 à 3 décrites plus haut. Ce programme est appelé par **selim**, mais il peut aussi être utilisé manuellement pour recréer un arbre des distances ou recalculer les étiquettes. Les résultats de l'étape 3 se retrouvent dans un fichier texte que l'on appelle ici fichier couverture. Voir la prochaine section pour le format de ce fichier.

- **Sélection globale des clusters**

Usage: mincov fichier

La sélection globale des clusters de tout les modèles se fait en utilisant les informations contenues dans le fichier couverture. Ce fichier provient de la liste triée des résultats de `covers all`. Les paramètres du tri sont: sort -n +0 +1 +2 file > out.

- **Ajout à l'arbre des distances**

Usage: adapic picno [model]

Ce programme est utilisé par **selim** pour ajouter une image à tous les arbres des distances.

Si le numéro du modèle est spécifié, l'image est ajoutée à ce modèle seulement.

4.2 Outils de vérification

Dans un projet de recherche en informatique, il est important de voir ce qui se passe à l'intérieur du système. Les programmes qui suivent furent développés de concert avec les programmes principaux. Certains de ces outils affichent en format texte les données binaires stockées dans les bases de données du système. D'autres en font un affichage graphique des données car ce mode de présentation donne souvent une meilleure idée de l'information contenue dans les données.

- **Impression de l'arbre des distances**

Usage: dumptree {-t n, -pno} [-gc] [-l] file

Options:

-t n	Totalise pour n clusters.
-pno	Fait une liste des feuilles de l'arbre.
-gc	Spécial pour les clusters générés.
-l	Format long, imprime plus de détails.

Ce programme va faire l'impression en format texte des informations contenues dans un arbre des distances. Ce programme est utilisé pour examiner les valeurs calculées pour chaque modèle. Notez que les fichiers d'arbre des distances se terminent par .zmt.

Exemple:

```
File Name: cohist.zmt
Revision Number: 0
Model Size: 3072
Number of nodes 64
Node: 64, 19.708920
| Node: 52, 9.844799
| | Node: 49, 4.920818
. . .
```

- **Fichier couverture**

Chaque cluster dans la solution est représentée par une ligne d'information dans un fichier de format texte. Le fichier doit être trié et ce format nous permet d'utiliser un programme de tri standard. De plus, le format texte facilite l'étude des résultats de l'algorithme. Voici le format d'une de ces lignes:

E U C u... c... M

E	Numéro de l'étiquette
U	Nombre d'imagettes spécifiées par l'usager
C	Nombre d'imagettes calculées
u...	Liste des imagettes spécifiées
c...	Liste des imagettes calculées
M	Numéro du modèle (pour information seulement)

- **Affichage de l'arbre des distances**

Usage: tree {-gc, -c, -lab, -pno, -pnd, -mod n} [-mag n] [-s] file

Options:

-gc	Format cercles pour clusters générés.
-c	Format pseudo-cercles.
-lab	Imprime le numéro des étiquettes (défaut).
-pno	Imprime le numéro des imagettes en octal.
-pnd	Imprime le numéro des imagettes en décimal.
-mod n	Imprime le numéro des imagettes modulo n.
-mag n	Facteur de grossissement.
-s	Tri des feuilles en ordre croissant.

Affiche l'arbre des distances en format graphique. Voir la Figure 2.8 pour un exemple de cette commande. Avec l'option -gc, l'arbre va être représenté en cercles concentriques et en points. Le rayon du cercle correspond au rayon dans l'arbre (voir Figure 3.7 dans la section 3.3).

- **Afficher une liste des modèles**

Usage: modlist

Ce programme va afficher une liste des modèles qui sont présentement dans le système.

- **Afficher une liste des images**

Usage: piclist

Ce programme va afficher une liste des images qui sont présentement dans le base des données.

- **Statistiques**

Usage: stats {1,2} [-t model_number]

Ce programme va créer des statistiques sur la précision de chaque modèle (voir section résultats). Aucun nom de fichiers n'est nécessaire car ce programme travaille directement sur la base de données **selim**. Pour utiliser ce programme les images d'une base doivent être complètement étiquetées. La commande `stats 1` va prendre une copie des étiquettes et en effacer la moitié de la base. De nouvelles étiquettes sont ensuite générées par `selim`. Finalement, la commande `stats 2` va générer des statistiques. L'option -t génère des statistiques dans un format utilisable par TEX et LATEX.

4.3 Détails d'implémentation

Cette section s'adresse à ceux qui veulent en savoir plus long sur l'implémentation du système et à ceux qui voudraient y apporter des améliorations. Un projet comme celui-ci devrait être fait au moins deux fois. La première version nous aide à comprendre le problème et nous donne l'occasion de trouver les points faibles de notre implémentation. À la suite de la première version nous sommes devenus experts dans le domaine et bien placés pour créer une deuxième version qui sera de beaucoup supérieure à la première.

4.3.1 Interface graphique

L'une des premières décisions prises fut le choix d'Amulet [23] comme interface graphique. Cette interface graphique, développée par l'université Carnegie Mellon, semblait prometteuse et certains membres de notre groupe de recherche l'utilisaient déjà [17].

Avantages

- Portabilité: les programmes qui utilisent Amulet peuvent être compilés sur différentes plateformes: X windows, Windows95/NT, Macintosh, etc.
- Extensibilité: de par sa nature, les objets Amulet peuvent facilement être modifiés ou combinés pour former de nouveaux objets.

Désavantages

- N'est pas un standard (difficile à maintenir).
- Ne peut lire que les images GIF et ne gère pas la palette des couleurs.
- Très lent à compiler, gros fichier exécutable.
- Le système n'est plus supporté.

4.3.2 Conception orientée objets

Quand les objets de base d'un système sont faciles à identifier la programmation orientée objet devient un choix naturel. Dans ce projet nous pouvons facilement identifier

plusieurs des objets: images, étiquettes, clusters, arbre des distances, etc. L'interface graphique Amulet est elle-même complètement de conception orientée objets. Voici une liste partielle des classes d'objets que nous avons créées pour ce projet:

ZIM: Fait la lecture d'un fichier image en mémoire et permet d'en accéder les composantes.

PICDB: Fait la gestion de la base de données **SELIM** .

LABELS: Fait la gestion des étiquettes spécifiées par l'usager.

DISTREE: Peut créer, sauvegarder ou lire sur disque un arbre des distances.

GENCLUST: Génère des clusters artificiels.

NUMVEC: Fait la gestion de certains vecteurs.

M2D: Facilite l'utilisation de matrices.

4.3.3 Architecture du système

Il est maintenant reconnu en génie logiciel qu'une bonne conception d'un système doit être modulaire. Si le système est aussi un outil de recherche, la conception modulaire n'en est que plus importante car nous devons pouvoir facilement changer un module sans affecter le reste. Le système d'étiquetage d'images est donc composé de plusieurs programmes et fichiers.

Nous pouvons voir à la Figure 4.3 que le système est composé de modules indépendants. Ces modules communiquent entre eux à l'aide de fichiers. Le module **selim** s'occupe de l'interface graphique et fait appel aux autres modules pour accomplir l'étiquetage. Le

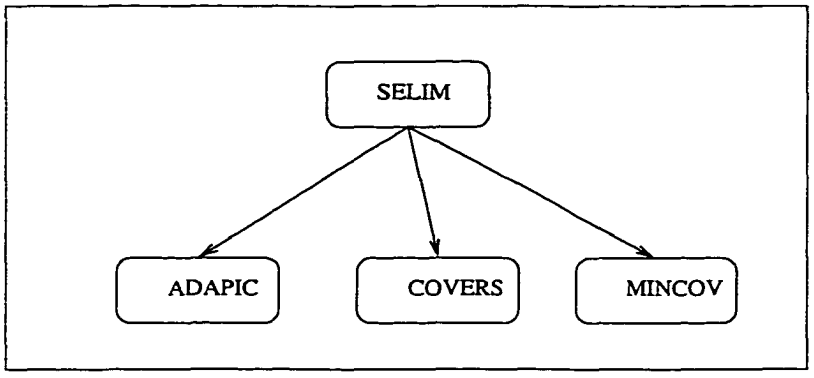


Figure 4.3: Architecture du système.

module **adapic** ajoute une image à la base de données, le module **covers** nous donne la liste des clusters et le module **mincov** fait le choix des meilleurs clusters.

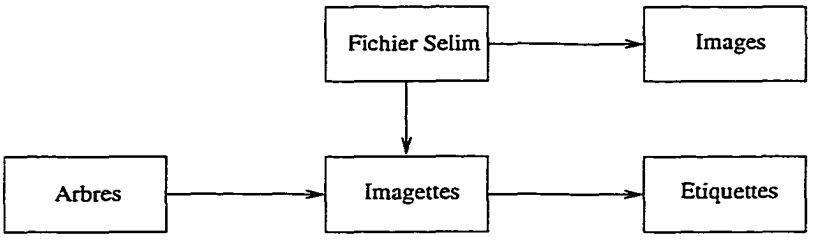


Figure 4.4: Relation entre les fichiers.

Les données temporaires et permanentes utilisées par le système sont stockées dans plusieurs fichiers. La Figure 4.4 nous montre la relation qui existe entre ces fichiers. Ces fichiers forment entre eux la base de données **SELIM** et se retrouvent tous dans le même répertoire à l'exception des images qui devraient se retrouver dans leur propre répertoire. Le fichier **SELIM** n'est en sorte qu'une liste de pointeurs vers les données contenues dans les autre fichiers. Il faut noter qu'il existe plusieurs fichiers arbres car il y a un arbre pour chaque type de modèle implémenté dans le système.

4.4 Résultats

Nous avons mentionné au début du chapitre deux qu'une étude sur différents systèmes de classification donnait des résultats dans les 80%. L'un des systèmes utilisés dans cette étude, les voisins les plus proches, s'apparente aux voisins-communs. Sa performance en moyenne est de 85% si les données utilisées pour les tests sont différentes des données utilisées à l'entraînement et une moyenne de 92% si les mêmes données sont utilisées pour l'entraînement et les tests.

Dans les résultats qui suivent nous avons pris soin de séparer les données d'entrainement et les données test. Les résultats ne sont donc que pour les données test. Les tests ont été faits à petite échelle pour montrer les détails du fonctionnement de l'étiquetage, et à grande échelle pour calculer la précision de l'étiquetage.

4.4.1 Petite échelle

Les résultats qui suivent furent calculés à partir de 4 images dont chacune est subdivisée en 64 imagettes. Nous avons divisé les résultats en différentes catégories pour analyser la contribution des différents éléments du système. Il est à noter que certaines imagettes sont difficiles à étiqueter. Par exemple, certaines sont mi-ciel, mi-arbre ou mi-eau, mi-plage. Néanmoins, les résultats semblent prometteurs.

- **Succès par modèle**

Nous voulons analyser ici la contribution de chaque modèle et le résultat de la combinaison

des modèles. Le Tableau 4.1 nous montre, pour chaque modèle, le nombre d'éléments du vecteur de caractéristiques, le pourcentage d'imagettes non-étiquetées et le pourcentage d'erreurs.

Modèle	grandeur	non-étiquetées	erreurs
Hasard	1	89.0	41.8
Uniformité	1	72.4	24.0
Couleur moyenne	6	70.4	2.7
Histogramme de teinte	360	38.2	2.6
Histogramme couleur	768	32.0	0.0
Tout les modèles	-	21.8	5.1

Tableau 4.1: Succès par modèle.

Le premier modèle, appelé *hasard*, ne calcule pas de caractéristiques, mais génère des points au hasard dans l'espace vectoriel du modèle. Le modèle *hasard* nous montre ce qui arrive si le modèle n'est pas bon. Le système ne peut créer de regroupement et la plupart des points demeurent sans étiquettes. Nous pouvons voir que le modèle *uniformité* n'est pas très bon. Ce modèle n'est composé que d'une des caractéristiques de la matrice de co-occurence. Ce modèle n'est pas assez riche. La couleur moyenne, qui utilise 6 nombres, est un peu mieux. Il semble que plus le vecteur est grand, plus les résultats sont bons.

Quand tous les modèles sont combinés nous pouvons voir que le pourcentage d'erreurs a quelque peu augmenté, mais que le pourcentage des imagettes non-étiquetées a diminué de beaucoup. C'est un bon résultat si nous considérons qu'un léger pourcentage d'erreurs est acceptable. De toute façon, ces résultats ne sont que des indications et il faut étudier les tests à grande échelle pour connaitre la précision du système.

- **Succès par étiquette**

Le tableau précédent, en donnant des résultats globaux, laisse de coté un point important. Le but du système est de sélectionner les meilleurs clusters de chaque modèle, et ce *pour chaque étiquette.* Le Tableau 4.2 nous donne, pour chaque modèle, le pourcentage de succès pour chaque étiquette.

	1	2	3	4	5	6	7	8	9
Hist. Couleur	87.5	68.8	66.7	42.9	35.3	0.0	88.2	93.8	0.0
Couleur moyenne	31.2	12.5	33.3	28.6	29.4	0.0	29.4	37.5	50.0
Hist. teinte	71.9	31.2	50.0	0.0	82.4	0.0	94.1	81.2	0.0
Uniformité	40.6	18.8	25.0	0.0	23.5	0.0	5.9	50.0	0.0

Tableau 4.2: Succès par étiquette.

Les 9 étiquettes sont:

1	ciel	4	arbres	7	terre
2	montagne	5	eau	8	gazon
3	herbe	6	forêt	9	plantes

Chaque nombre représente le succès du modèle vis-à-vis une étiquette spécifique. Par exemple, l'histogramme couleur reconnait correctement 87.5% des imagettes ciel. Nous pouvons donc voir ici les forces et les faiblesses de chaque modèle. Nul modèle n'a reconnu la forêt car ses caractéristiques ressemblaient trop à celles des plantes. Pour augmenter la performance du système, il faudrait donc créer un nouveau modèle qui prendrait en considération les différences entre la forêt et les plantes.

4.4.2 Grande échelle

L'article du MIT que nous utilisons comme comparaison [27] cite un taux de réussite de 90% dans un test d'une centaine d'images. Cependant, 80% des imagettes dans ce test ne furent pas étiquetées. Nous avons récupéré les images utilisées dans ces tests[5] pour y comparer notre propre système. Ce sont des photos *de voyage* qui sont plus ou moins nettes et qui sont très variées. Ces photos contiennent des scènes d'intérieur et d'extérieur, de ville et de campagne, de personnes et de foules, etc.

Nous avons fait nos tests en deux étapes. Dans la première étape qui est la phase d'apprentissage, nous avons visionné chaque image pour y ajouter quelques étiquettes. Nous avons ainsi ajouté environ 250 étiquettes à la base de données. Dans la deuxième étape nous avons fait un traitement par lots pour étiqueter toutes les images puis nous les avons visionnés de nouveau pour noter les erreurs sur papier. Il est certain qu'en mode entièrement interactif, en "jouant" avec l'ajout des étiquettes, nous aurions obtenu de meilleurs résultats.

Dans le Tableau 4.3 nous avons reporté à la première ligne les résultats du test à petite échelle. La deuxième ligne nous donnent les résultats du test avec les images du MIT. Enfin, la troisième ligne affiche les résultats publiés par l'équipe du MIT. La deuxième colonne du tableau donne le pourcentage des imagettes qui ne furent pas étiquetées. La troisième colonne indique le pourcentage d'erreurs pour les imagettes étiquetées par le système.

[5]Nous remercions l'équipe du MIT pour nous avoir donné accès à ces images.

Modèle	non-étiquetées	erreurs
Petite échelle	21.8	5.1
Grande échelle	74.6	12.5
Résultat du MIT	80	10

Tableau 4.3: Test à grande échelle.

À première vue nos résultats semblent comparables à ceux de l'équipe du MIT. Nous avons étiqueté un peu plus d'images, mais nous avons un peu plus d'erreurs. Il est cependant difficile de comparer ces deux résultats car trop de variables restent incontrolées. Nous avons certainement les mêmes images, mais nous n'avons pas le même étiquetage. De plus, comme l'étiquetage est en partie subjectif, les résultats ne seront pas les mêmes pour deux individus.

Deux points importants soulignent l'aspect subjectif des résultats: le grand nombre d'imagettes qui restent non-étiquetées (75% à 80%) et la distribution des erreurs d'étiquetages. Le Tableau 4.4 qui nous montre le pourcentage d'images par rapport aux erreurs. La deuxième colonne indique le pourcentage des images qui ont la quantité d'erreurs indiquée à la colonne un. Nous voyons que la majorité des images ont zéro ou une erreur. Donc, quand les erreurs apparaissent, elles sont assez nombreuses pour être visibles à l'utilisateur qui peut faire les corrections qui s'imposent en ajoutant une ou deux étiquettes.

Nombre d'erreurs	% des images
0	38
1	30
2-5	20
>5	12

Tableau 4.4: Distribution des erreurs.

4.5 Conclusion

Dans la première partie de ce chapitre nous avons décrit le système d'étiquetage d'images tel que conçu. Cette description permet à un usager d'avoir une idée du fonctionnement du système et surtout de lui montrer comment utiliser le système pour faire de l'étiquetage. Dans la deuxième partie, nous avons décrit les résultats obtenus par le système sur des cas réels. Le système montre une certaine utilité en mode interactif, mais n'est pas encore prêt pour le traitement par lots. En effet, même si les deux tiers des images sont étiquetées sans ou avec une erreur, un petit pourcentage d'images se retrouvent avec un nombre d'erreurs qui rend leurs étiquetages douteux.

CONCLUSION

Le projet qui fait l'objet de ce mémoire avait pour but la création d'un système pour l'étiquetage automatique des images. Autant que possible, le système devait paraître professionnel, être rapide et donner de bons résultats. Ce mémoire lui-même avait deux buts: décrire les étapes suivies pour atteindre les objectifs du système d'étiquetage et expliquer le pourquoi de chaque décision.

Pour paraître professionnel, le système devait se conformer aux normes établies dans le domaine informatique. L'utilisation d'une interface graphique est maintenant devenue un prérequis que nous avons satisfait en utilisant Amulet. Notre système est de conception modulaire, ce qui en facilite la programmation et l'entretien. Enfin, en plus du mode interactif, le système peut fonctionner en traitement par lots.

L'objectif de vitesse fut atteint grâce à un nouvel algorithme de clustering qui fonctionne de façon incrémentale. Ce nouvel algorithme, que nous avons nommé arbre des distances, est une contribution originale de ce travail. Grâce à cet algorithme, ajouter une nouvelle image au système d'étiquetage est un processus qui prend moins de 10 secondes. Avec l'algorithme de Jarvis-Patrick, le même travail prendrait près d'une demi-heure.

Avec l'utilisation de l'excellente idée d'une équipe du MIT, à savoir la combinaison de modèles multiples, nous nous sommes assurés de résultats qui sont supérieurs à l'utilisation de modèles individuels. Le système d'étiquetage ne fonctionne pas sans erreur, mais ce système n'est qu'une étape dans un processus de classification de l'image. Un système de classification pourrait rejeter l'étiquette qui n'apparaît qu'une fois dans une image ou qui semble suspecte dans le contexte des autres étiquettes de l'image.

Vers la fin de ce projet, nous avons appris que l'interface graphique Amulet ne serait plus supportée et que la version en développement était abandonnée. Ceci nous pose un problème car plusieurs des extensions projetées nécessitaient des options qu'Amulet avait annoncées dans leur prochaine version. Il faudra donc, à la reprise de ce projet, refaire l'interface graphique et utiliser une interface plus standard.

Afin d'améliorer l'exactitude du système, plusieurs nouveaux projets pourraient être entrepris. Une segmentation de l'image au lieu d'un découpage en quadrillé augmenterait de beaucoup l'exactitude du système. Le système fonctionnerait mieux avec une série de modèles qui se complémentent l'un l'autre. L'algorithme de clustering qui fonctionne maintenant de façon binaire devrait être refait en arbre n-aire. Finalement, l'implémentation d'un système de classification qui utilise les résultats de l'étiquetage ferait un bon *démonstrateur* pour le système d'étiquetage.

BIBLIOGRAPHIE

[1] Michael R. Anderberg. *Cluster analysis for applications.* Academic Press, 1973.

[2] Noureddine Abbadeni, Djemel Ziou, Shengrui Wang. *Recherche d'images basée sur le contenu. Représentation de la texture par le modèle autorégressif.* Rapport technique No. 216, Université de Sherbrooke, 1998.

[3] Glenn Becker. *Content-based Query of Image Database.* Thomson Technology Services Group (TTSG), 1996.

[4] John Bradley. *XV: Interactive Image Display for the X Window System.* 1989-1994.

[5] Gilles Daigle, Shengrui Wang, Djemel Ziou. *Recherche d'images basée sur le contenu.* Rapport technique No. 212, Université de Sherbrooke, 1998.

[6] Gilles Daigle, Shengrui Wang, Djemel Ziou, Béchir El Ayeb. *A System for Picture Labeling.* IEEE International Conference on Systems, Man, and Cybernetics, 1998, pp. 4453-4458.

[7] Geoff M. Downs, John M. Barnard. *Fast Clustering of Very Large Datasets and Combinatorial Libraries.* Conference on Computational Approaches to the Design and Analysis of Combinatorial Libraries, April 14-16, 1998.

[8] M. Flickner, H. Sawhney, W. Niblack et al. *Query by Image and Video Content: The QBIC System.* IEEE Computer, Vol. 28, N. 9, pp. 23-32, Sept 1995.

[9] Borko Furht, Stephen W. Smoliar, HongJian Zhang. *Video and image processing in multimedia systems.* Kluwer Academic Publishers, 1995.

[10] Monika M. Gorkani, Rosalind W. Picard. *Texture Orientation for sorting photos "at a glance".* MIT Media Laboratory, Technical Report No. 292 and IEEE Conference on Pattern Recognition, Jerusalem, Oct. 1994.

[11] S. Geffner, D. Agrawal, A. El Abbadi, T. Smith, M. Larsgaard. *Smart Indexes for Efficient Browsing of Library Collections.* IEEE Forum on Research and technology advances in digital libraries (IEEE ADL'98), 1998, pp. 107-116.

[12] Phil Greenway. *Finding Roads in Images.* Mobile Robots IX, Proceedings of the SPIE vol 352, 1995.

[13] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision.* Addison-Wesley, 1992.

[14] Donald Hearn, M. Pauline Baker. *Computer Graphics.* Prentice Hall Inc, Englewood Cliffs, NJ, 1986.

[15] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall Inc, Englewood Cliffs, NJ, 1988.

[16] R. A. Jarvis, Edward A. Patrick. *Clustering Using a Similar Measure Based on Shared Near Neighbors. IEEE Trans. Comput.*, 1973, C-22, 1025-1034.

[17] Hatem Jedidi. *Outils logiciels pour la surveillance et la détection des fautes dans les systèmes distribués.* Mémoire de maîtrise, Université de Sherbrooke, 1997.

[18] Donald E. Knuth. *The art of computer programming.* Addison-Wesley, 1969.

[19] B. S. Manjunath, W. Y. Ma. *Texture Features for Browsing and Retrieval of Image Data.* IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol. 18, No. 8, August 1996.

[20] T. P. Minka and Rosalind W. Picard. *Interactive learning using a "society of models".* MIT Media Laboratory, Technical Report No. 349, 1995.

[21] W. E. Moen, E. L. Stewart. *Assessing Metadata Quality: Findings and Methodological Considerations from an Evaluation of the U.S. Goverment Information Locator Service (GILS).* IEEE Forum on Research and technology advances in digital libraries (IEEE ADL'98), 1998, pp. 246-255.

[22] Fionn Murtagh. *Multidimensional Clustering Algorithms.* Compstat Lectures 4, Physica-Verlag, 1985.

[23] Brad A. Myers. *The Amulet V3.0 Reference Manual.* Carnegie Mellon University, 1997.

[24] Dinesh Nair, Amar Mitiche, J. K. Aggarwal. *On comparing the performance of object recognition systems.* IEEE International Conference on Image Processing, Vol 2, 1995.

[25] Tan Bao Nguyen. *Évaluation des algorithmes d'extraction dans les images à niveaux de gris.* Mémoire de maîtrise, Université de Sherbrooke, 1998.

[26] A. Pentland, Rosalind W. Picard, S. Sclaroff. *Photobook: Content-Based Manipulation of Image Databases.* International Journal of Computer Vision, 18(3), pp. 233-254, 1996.

[27] Rosalind W. Picard and T. P. Minka. *Vision Texture for Annotation.* MIT Media Laboratory, Technical Report No. 302, 1995.

[28] A. Ravishankar Rao. *A Taxonomy fot Texture Description and Identification.* Springer-Verlag, 1990.

[29] Paul Robert. *Le petit ROBERT, dictionnaire de la langue française.* Société du nouveau Litrée, 1976.

[30] Hanan Samet, Aya Soffer. *MARCO: MAp Retrieval by COntent.* IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol. 18, No. 8, August 1996.

[31] Daniel S. Swets and Juyang Weng. *Efficient Image Retrieval using a Network with Complex Neurons.* IEEE, 1995.

[32] Daniel S. Swets and Juyang Weng. *Using Discriminant Eigenfeatures for Image Retrieval.* IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol. 18, No. 8, August 1996.

[33] Martin Szummer, Rosalind W. Picard. *Indoor-Outdoor Image Classification.* MIT Media Laboratory, Technical Report No. 445 and IEEE Intl Workshop on Content-Based Access of Image and Video Databases, Jan 1998.

[34] Hideyuki Tamura, Shunji Mori, Takashi Yamawaki. *Textural Features Corresponding to Visual Perception.* IEEE International Conference on Systems, Man, and Cybernetics, Vol. 8, No. 6, Jan 1978.

[35] Fumiaki Tomita, Saburo Tsuji. *Computer Analysis of Visual Textures.* Kluwer Academic Publishers, 1990.